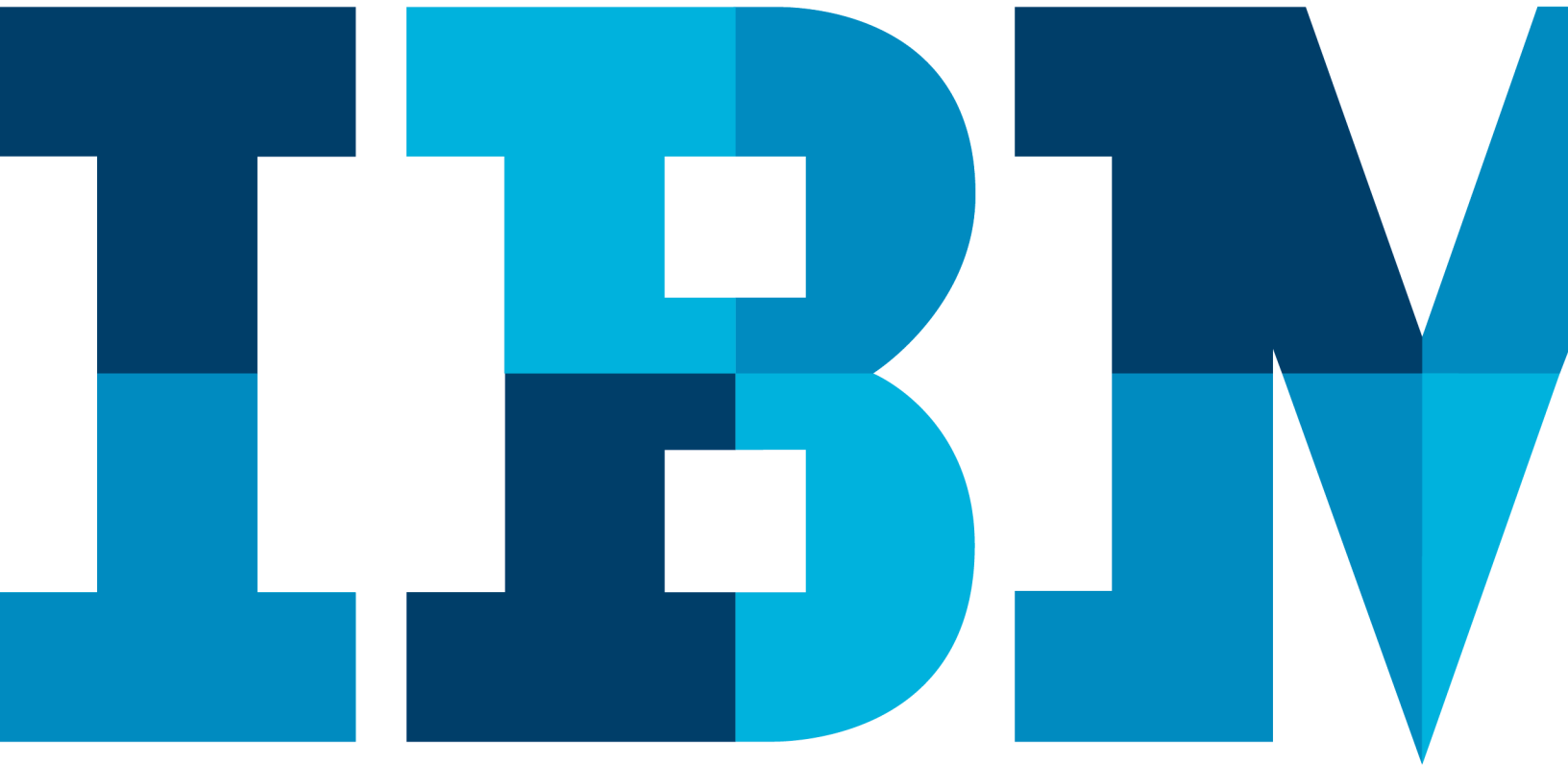


# IBM Blockchain Hands-On Composer SDK

*Lab Four*



# Contents

<b>CONTENTS</b>	<b>2</b>	
<b>1</b>	<b>OVERVIEW .....</b>	<b>3</b>
<b>2</b>	<b>INSTALLATION.....</b>	<b>4</b>
2.1	PREREQUISITE .....	4
2.2	INSTALL THE COMPOSER CLIENT ON YOUR WORKSTATION.....	4
2.3	CREATE A FOLDER FOR THE PROJECT .....	4
2.4	CONFIGURE THE REMOTE ACCESS TO THE IBM BLOCKCHAIN STARTER PLAN .....	4
2.4.1	RETRIEVE THE CONNECTION PROFILE FROM THE STARTER PLAN.....	4
2.4.2	RETRIEVE THE SECRET IN THE CONNECTION PROFILE FILE.....	4
2.4.3	CREATE A CERTIFICATE AUTHORITY CARD AND ADD THE CERTIFICATE TO THE STARTER PLAN ..	5
2.4.4	CREATING AN ADMIN BUSINESS NETWORK CARD .....	7
2.5	CREATE THE BUSINESS NETWORK (MARBLE-NETWORK).....	8
2.5.1	CREATE THE BUSINESS NETWORK (MARBLE-NETWORK) WITH COMPOSER-PLAYGROUND.....	8
2.5.2	CREATE THE BUSINESS NETWORK (MARBLE-NETWORK) WITH COMPOSER-CLI .....	10
2.6	DEPLOY THE BUSINESS NETWORK .....	14
<b>3</b>	<b>MANIPULATING AND ADDING RESOURCES WITH THE SDK.....</b>	<b>16</b>
3.1	GETTING STARTED .....	16
3.2	CREATE THE PARTICIPANTS AND ASSETS .....	16
3.3	GET AND DISPLAY THE LIST OF MARBLES .....	18
<b>4</b>	<b>USING THE COMPOSER REST SERVER .....</b>	<b>19</b>
4.1	INSTALL THE COMPOSER REST SERVER ON IBM CLOUD .....	19
4.2	CHECK THE REST API .....	20
4.2.1	DISPLAY A MARBLE.....	21
4.2.2	EXECUTE A TRANSACTION .....	22
<b>NOTICES</b>	<b>24</b>	
<b>APPENDIX A.</b>	<b>TRADEMARKS AND COPYRIGHTS.....</b>	<b>26</b>

---

# 1 Overview

The aim of this lab is to get you familiar with developing Hyperledger Composer business networks. We will do this by exploring the Composer modelling language, how to write transaction processor functions in JavaScript and lastly examine how Access Control is managed in Composer.

The lab will also familiarize you with the Composer Playground, a web-based tool that allows for rapid development and testing of Composer business networks.

It should be noted that while the contents of this lab will predominantly occur within Composer Playground (for the sake of accelerating the learning and development process), the Lab can easily be completed offline and using a text editor such as Visual Studio Code or Atom. In this case please refer to the next Lab for instructions on how to use the command line tools available in Composer.

## 2 Installation

### 2.1 Prerequisite

This lab requires to have an IBM Blockchain Starter Plan service subscribed on IBM Cloud.

Install the Nodejs Package Manager npm.

Then initialize your environment :

```
npm init
```

We recommend using Visual Studio Code as text editor to write the code of your Blockchain application (smartcontract as well as client application)

### 2.2 Install the Composer client on your workstation

You have to install the composer client to deploy the Business Network Archive and interact with the composer server in IBM Cloud.

Open a Linux terminal and issue the following command

```
npm install -g composer-cli@0.19.5
```

(on MacOS you have to run the command as root using sudo and adding parameters:  
sudo npm install -g composer-cli@0.19.5 --unsafe-perm=true --allow-root)

### 2.3 Create a folder for the project

```
cd ~/git  
mkdir marble-project  
cd marble-project
```

### 2.4 Configure the remote access to the IBM Blockchain Starter Plan

#### 2.4.1 Retrieve the connection profile from the starter plan

Go to the Overview windows of the IBSP, then click on connection profile button and choose download. Then move the file in the marble-project folder, in the file connection-profile.json:

Go to the download folder and run the command :

```
cd ~/Downloads  
mv <creds_..._org1.json> ~/git/marble-project/connection-profile.json
```

#### 2.4.2 Retrieve the secret in the connection profile file

Display the content of the connection-profile :

```
cat ~/git/marble-project/connection-profile.json
```

at the end of the file, under the “certificate authorities” section, (after “registrar”) take the “enrolSecret value (the following extract is an example of connection profile):

```

    "certificateAuthorities": {
      "org1-ca": {
        "url": "https://n846c6d511f81498eb91783a02ceef5dc-org1-ca.uk02.blockchain.ibm.com:31011",
        "httpOptions": {
          "verify": true
        },
        "tlsCACerts": {
          "pem": "-----BEGIN CERTIFICATE...-----END CERTIFICATE-----\r\n"
        },
        "registrar": [
          {
            "enrollId": "admin",
            "enrollSecret": "464bfa9a26",
            "x-affiliations": [
              "org1",
              "org1.department1",
              "org1.department2",
              "org2",
              "org2.department1"
            ]
          }
        ],
        "caName": "org1CA",
        "x-mspid": "org1"
      }
    },
  },
}

```

### 2.4.3 Create a certificate authority card and add the certificate to the starter plan

Move to the folder of the connection profile file, then create the certificate authority card using the composer client command and the enrollSecret value retrieved in the step before:

```

cd ~/git/marble-project
composer card create -f ca.card -p connection-profile.json -u admin -s <enrollSecret>

```

Import the card using the following command (this will import the card in your local composer environment):

```

composer card import -f ca.card -c ca

```

Then retrieve the certificates for the ca (they will be put in the credentials folder of your current folder) using the following command (pay attention to replace <enrollSecret> by the enrollSecret value retrieved previously):

```

composer identity request --card ca --path ./credentials -u admin -s <enrollSecret>

```

Display the public key of the admin

```

cat credentials/admin-pub.pem

```

It should display a certificate similar to:

```

-----BEGIN CERTIFICATE-----
MIIB8TCCAzigAwIBAgIUUQyCGAl31oeSW6Hfk1B8gauhGz0wCgYIKoZIzj0EAwIw
.../PMwCgYIKoZIzj0EAwIDRwAw
RAIgBRj2WRb5bejLOAyf18FhdqUv66uezqkvjWTaNv2M6o4CIFrrdSRXtXyr1Dj6
GOxKM7xRnMGtbXKnK8tOmsVdztTt

```

-----END CERTIFICATE-----

Then copy this certificate content, go to the Starter Plan, click on Member menu, and select the tab Certificates

The screenshot shows the IBM Blockchain Starter Plan interface. The top navigation bar includes 'IBM Blockchain Starter Plan', 'Cookie Preferences', and a user profile 'OLIVIER VALLOD' with a dropdown arrow. The left sidebar is titled 'Olympus Network - JRQ' and contains a 'MY NETWORK' section with links: Overview, Members (highlighted), Channels, Notifications, Certificate Authority, APIs, and MY CODE. Below MY CODE are links for 'Develop code' and 'Install code'. The main content area is titled 'Members' and includes a description: 'View and manage the members (organizations) of the network on the Members tab. You can invite other organizations to the network or add more organizations in addition to the two organizations that you own by default. You can also view and manage admin certificates that are associated to your organizations on the Certificates tab. [Learn more](#)'. There are two tabs: 'Members' and 'Certificates' (selected). A blue 'Add Certificate' button is in the top right. Below is a table with columns 'NAME', 'DATE ADDED', and 'ACTION'. The table is empty, and a message 'No Certificates Found' is displayed at the bottom.

Click on the button Add Certificate, then fill in cert1 as name, and past the certificate content in the field Certificate.

Then it will propose to restart the peers: click on Restart:

This screenshot shows the same interface as before, but with a 'Restart peers' dialog box open. The dialog has a title bar with a close button (X). The main text inside says: 'The certificate is uploaded successfully, you need to restart all your peers before submitting transactions. Do you want to restart your peers?'. At the bottom of the dialog are two buttons: 'Cancel' and 'Restart'. The background interface is dimmed, showing the 'Members' tab and the 'No Certificates Found' message.

Then click on the menu Channels, on the line of the defaultchannel, click on the actions button, then select the menu Sync Certificates, then click on Submit

## 2.4.4 Creating an admin business network card

Now that the correct certificates have been synced with the peers, business network cards can be created which have the permissions to install the Hyperledger Composer runtime and start chaincode.

Create an admin card with the channel admin and peer admin roles by using the following command:

```
cd ~/git/marble-project
composer card create -f adminCard.card -p connection-profile.json -u admin -c
./credentials/admin-pub.pem -k ./credentials/admin-priv.pem --role PeerAdmin --role
ChannelAdmin
```

This card will be used to deploy a business network to Starter Plan.

Import the card created in the previous step using the following command:

```
composer card import -f adminCard.card -c adminCard
```

To check that the card is well imported, issue the following command

```
composer card list
```

Expected result:

**The following Business Network Cards are available:**

**Connection Profile:** Olympus Network – jRQ

Card Name	UserId	Business Network
ca	admin	
adminCard	admin	

```
|_____|_____|_____|
```

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

Command succeeded

## 2.5 Create the Business Network (marble-network)

In the Lab 3, we have seen that the Composer Playground allows to create a Business Network and test it. This business network can be exported as a Business Network Archive (.bna). This archive file can also be generated from flat file using the composer command. We are describing both approaches here.

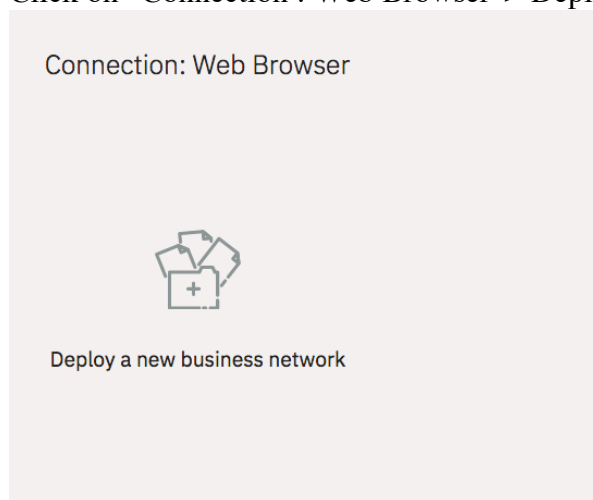
### 2.5.1 Create the Business Network (marble-network) with composer-playground

Open the Composer Playground (using Firefox browser):

<YOUR PUBLIC IP ADDRESS>:31080

Or <https://composer-playground.mybluemix.net/login>

Click on “Connection : Web Browser”> Deploy a new business network



Then fill in the information (name : marble-network, network admin card : admin@ marble-network) and choose marbles-network as Business Network Definition. Then click on DEPLOY.



← My Wallet

Deploy New Business Network

**1. BASIC INFORMATION**

Give your new Business Network a name: marble-network

Describe what your Business Network will be used for: Marble business network

Give the network admin card that will be created a name: admin@marble-network

**2. MODEL NETWORK STARTER TEMPLATE**

Choose a Business Network Definition to start with:  
Choose a sample to play with, start a new project, or import your previous work

basic-sample-network empty-business-network Drop here to upload or browse

Samples on npm

animaltracking-network bond-network carauction-network digitalproperty-network fund-clearing-network letters-of-credit-network **marbles-network** perishable-network



On the My business Networks page, click on Connect now on the marble-network.

Connection: Web Browser

**admin@marble-network**

USER ID  
admin

BUSINESS NETWORK  
marble-network

[Connect now →](#)

Update the Model.cto, and logic.js (retrieve the content given at the chapter 2.4.2).  
Then click on Deploy Changes.

Finally, click on export to retrieve the Business Network Archive (.bna) that you will use to deploy on the Hyperledger Fabric in IBM Cloud.

The screenshot shows the 'Define' tab of the IBM Blockchain Composer Playground. The interface is divided into a left sidebar and a main content area.

**Left Sidebar:**

- Web:** marble-network
- Define:** (Active tab)
- Test:**
- FILES:**
  - About:** README.md, package.json
  - Model File:** models/marbles.cto
  - Script File:** lib/logic.js
  - Access Control:** permissions.acl
- UPDATE NETWORK:**
  - From: 0.2.6-20180530153450
  - To: 0.2.6-deploy.0
  - Deploy changes** (button)

**Main Content Area:**

**About File README.md**

### Marbles Network

*This is an interactive and distributed, marble trading demo. List marbles for sale and exchange marbles between participants.*

This business network defines:

- Participant:** Player
- Asset:** Marble
- Transaction:** TradeMarble

Player participants are able to have Marble assets and trade these with TradeMarble transaction.

To test this Business Network Definition in the **Test** tab:

Create two Player participant:

```

{
  "$class": "org.hyperledger_composer.marbles.Player",
  "email": "memberA@acme.org",
  "firstName": "Jenny",
  "lastName": "Jones"
}

{
  "$class": "org.hyperledger_composer.marbles.Player",
  "email": "memberB@acme.org",
  "firstName": "Billy",
  "lastName": "Thompson"
}

```

### 2.5.2 Create the Business Network (marble-network) with composer-cli

Here we are describing the second approach to create a Business Network Archive file : it consists in using the composer-cli (instead of the Composer Playground). We are creating manually the different files that constitutes the BNA - the package description, the ACL, the model, and the logic file. Then we are generating the BNA file.

Create a folder called **marbles-network**.

```
cd ~/git
mkdir marbles-network
cd marbles-network
```

Create a file called **package.json** in this folder, and put the following content.

```
{
  "engines": {
    "composer": "^0.19.5"
  },
  "name": "marbles-network",
  "version": "0.1.1-deploy.0",
```

```

"description": "Marble Trading Network",
"scripts": {
  "prepublish": "mkdirp ./dist && composer archive create --sourceType dir --sourceName
. -a ./dist/marbles-network.bna",
  "pretest": "npm run lint",
  "lint": "eslint .",
  "postlint": "npm run licchk",
  "licchk": "license-check-and-add",
  "postlicchk": "npm run doc",
  "doc": "jsdoc --pedantic --recurse -c jsdoc.json",
  "test": "mocha -t 0 --recursive",
  "deploy": "./scripts/deploy.sh"
},
"repository": {
  "type": "git",
  "url": "https://github.com/hyperledger/composer-sample-networks.git"
},
"keywords": [
  "marbles",
  "trading",
  "composer",
  "composer-network"
],
"author": "Hyperledger Composer",
"license": "Apache-2.0",
"devDependencies": {
  "chai": "^3.5.0",
  "composer-admin": "^0.19.5",
  "composer-cli": "^0.19.5",
  "composer-client": "^0.19.5",
  "composer-common": "^0.19.5",
  "composer-connector-embedded": "^0.19.5",
  "eslint": "^3.6.1",
  "istanbul": "^0.4.5",
  "jsdoc": "^3.5.5",
  "license-check-and-add": "~2.3.0",
  "mkdirp": "^0.5.1",
  "mocha": "^3.2.0",
  "moment": "^2.17.1"
}
}

```

Create a file called permissions.acl

```

/**
 * Sample access control list.
 */
rule Default {
  description: "Allow all participants access to all resources"
  participant: "ANY"
  operation: ALL
}

```

```

    resource: "org.hyperledger_composer.marbles.*"
    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
}

rule NetworkAdminUser {
  description: "Grant business network administrators full access to user resources"
  participant: "org.hyperledger.composer.system.NetworkAdmin"
  operation: ALL
  resource: "*"
  action: ALLOW
}

rule NetworkAdminSystem {
  description: "Grant business network administrators full access to system resources"
  participant: "org.hyperledger.composer.system.NetworkAdmin"
  operation: ALL
  resource: "org.hyperledger.composer.system.*"
  action: ALLOW
}

```

Create a subfolder called **models**.

```

mkdir models
cd models

```

Create a file called **marble.cto** in this folder, and put the following content. It consists of the assets, participants and transactions of the Business Network.

```

/**
 * Defines a data model for a marble trading network
 */
namespace org.hyperledger_composer.marbles

enum MarbleColor {
  o RED
  o GREEN
  o BLUE
  o PURPLE
  o ORANGE
}

enum MarbleSize {
  o SMALL
  o MEDIUM
  o LARGE
}

```

```

}
asset Marble identified by marbleId {
  o String marbleId
  o MarbleSize size
  o MarbleColor color
  --> Player owner
}
participant Player identified by email {
  o String email
  o String firstName
  o String lastName
}
transaction TradeMarble {
  --> Marble marble
  --> Player newOwner
}
transaction ChangeMarbleSize {
  --> Marble marble
  o MarbleSize newSize
}

```

Create a subfolder of marble-network called **lib**.

```

cd ..
mkdir lib
cd lib

```

Create a file called **logic.js** in this folder and put the following content. It consists of the code of the transactions.

```

/* global getAssetRegistry */

/**
 * Trade a marble to a new player
 * @param {org.hyperledger_composer.marbles.TradeMarble} tradeMarble - the trade marble
transaction
 * @transaction
 */
async function tradeMarble(tradeMarble) { // eslint-disable-line no-unused-vars
  tradeMarble.marble.owner = tradeMarble.newOwner;
  const assetRegistry = await getAssetRegistry('org.hyperledger_composer.marbles.Marble');
  await assetRegistry.update(tradeMarble.marble);
}

/**
 * Change the size of a Marble
 * @param {org.hyperledger_composer.marbles.ChangeMarbleSize} changeMarbleSize - the change
marble size transaction
 * @transaction
 */
async function changeMarbleSize(changeMarbleSize) { // eslint-disable-line no-unused-vars
  changeMarbleSize.marble.size = changeMarbleSize.newSize;
}

```

```
const assetRegistry = await getAssetRegistry('org.hyperledger_composer.marbles.Marble');
await assetRegistry.update(changeMarbleSize.marble);
}
```

### Now we are generating the Business Network Archive.

Go to the folder marble-network and run the following command :

```
cd ~/git/marble-project
composer archive create -t dir -n cd ~/git/marbles-network
```

You should get an error:

```
Input directory: /home/blockchain/git/marbles-network
SyntaxError: Failed to parse /home/blockchain/git/marbles-network/lib/logic.js:
Unexpected token (14:0)
```

Command failed

Fix it and redo the archive creation.

Expected result:

Creating Business Network Archive

Looking for package.json of Business Network Definition

Input directory: /Users/ovalldo/git/marbles-network

Found:

```
Description: Marble Trading Network
Name: marbles-network
Identifier: marbles-network@0.1.1-deploy.0
```

Written Business Network Definition Archive file to

Output file: marbles-network@0.1.1-deploy.0.bna

Command succeeded

## 2.6 Deploy the Business Network

We first install the business network based on the Business Network archive:

```
cd ~/git/marble-project
composer network install -a marbles-network@0.1.1-deploy.0.bna -c adminCard
```

✓ Installing business network. This may take a minute...

Successfully installed business network marbles-network, version 0.1.1-deploy.0

Command succeeded

Then we start the business network and generate the business network card for the administration of this BN:

```
composer network start -c adminCard -n marbles-network -V 0.1.1-deploy.0 -A admin -C
credentials/admin-pub.pem -f delete_me.card
```

Expected result:

Starting business network marbles-network at version 0.1.1-deploy.0

Processing these Network Admins:

userName: admin

✓ Starting business network definition. This may take a minute...

**Successfully created business network card:**

Filename: admin@marbles-network.card

Command succeeded

Now we create a Business Network card for the operator of the business network :

```
composer card create -n marbles-network -p connection-profile.json -u admin -c
credentials/admin-pub.pem -k credentials/admin-priv.pem
```

Finally, we import the BN card of the admin user:

```
composer card import --file admin@marbles-network.card
```

Ping the BN in order to check it is up running and to do the enrollment of the admin user:

```
composer network ping -c admin@marbles-network
```

Expected result:

**The connection to the network was successfully tested:** marbles-network

Business network version: 0.1.1-deploy.0

Composer runtime version: 0.19.5

participant: org.hyperledger.composer.system.NetworkAdmin#admin

identity:

org.hyperledger.composer.system.Identity#7c680cf85063a9f8b1082acd5e0a31daf856cf2fb6a8e55f0eb288529ba2218d

Command succeeded

At this stage, we have prepared the local environment (composer cli) to access to the remote environment (Hyperledger Fabric and Composer in a Kubernetes clusters in IBM Cloud - ex Bluemix -) and we have deployed the Business Network archive of the Marble Smartcontract.

## 3 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 to adding and updating resources.*

### 3.1 Getting Started

First, create a folder **marble-client** where the application files will be added.

```
cd ~/git
mkdir marble-client
cd marble-client
```

### 3.2 Create the participants and assets

In this part, we are creating a simple NodeJS application which will create participants and asset of the marble-network.

Use a text editor and create a file called add-resources.js

First we reference the NodeJS lib that we are using : composer-client, then we put the skeleton of our application : an asynchronous function 'createResources()' and the call to this function. The result (t) will be displayed ('console.log(t)').

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

async function createResources() {
  try {
    ...
  } catch (error) {
    console.log(error);
    process.exit(1);
  }
}

createResources().then((t) => {
  console.log(t);
});
```

Then in the core of the function, replace the '...' by the connection to the Business Network.

```
// Connect to the Business Network
let bizNetConnection = new BusinessNetworkConnection();
```



```
let bizNetDef = await bizNetConnection.connect("admin@marbles-network");
```

Then just after, retrieve the description of the Business Network.

```
// Retrieve the description of the Business Network
let factory = bizNetDef.getFactory();
```

Then we are creating the first resource which is a participant of type Player.

```
// Create a resource of type Player
let player1 = factory.newResource('org.hyperledger_composer.marbles', 'Player',
'email:olivier2@mele');
player1.lastName = 'Truc';
player1.firstName = 'Olivier2';
```

Create other Player resources, replicating this code and changing the values: create player2 and player3.

Then retrieve the participant registry of our BN ('org.hyperledger\_composer.marbles.Player') and add the created resources - player1, player2 and player3 - in this registry

```
//retrieve the participant registry and add the Player resources
let playerRegistry = await
bizNetConnection.getParticipantRegistry('org.hyperledger_composer.marbles.Player');
await playerRegistry.addAll([player1, player2, player3]);
```

At this stage, you can run this application which will create 3 Participants: in the folder, you can run the following command:

```
node add-resources.js
```

To check the added resources, use the command:

```
composer network list -c admin@marbles-network
```

Pay attention to comment the line “await playerRegistry.addAll ...” after running the application to avoid an error with duplicate resource. (You can also change the value of the emailId of each Player).

Now we are going to add the assets. Create a new resource of type Marble. Then assign the values to each field (size, color). Then, create a relationship with the selected Player giving his email.

```
let marble = factory.newResource('org.hyperledger_composer.marbles', 'Marble',
'marbleId:1');
marble.size = 'MEDIUM';
marble.color = 'ORANGE';
marble.owner = factory.newRelationship('org.hyperledger_composer.marbles', 'Player',
'email:olivier1@mele');
```

Create marble1 and marble2 resources replicating the previous code and changing the values.

Then retrieve the marble registry of our BN ('org.hyperledger\_composer.marbles.Marble') and add the created resources - marble, marble1 and marble2 - in this registry

```
let marbleRegistry = await
bizNetConnection.getAssetRegistry('org.hyperledger_composer.marbles.Marble');
await marbleRegistry.add(marble);
await marbleRegistry.add(marble1);
await marbleRegistry.add(marble2);
```

### 3.3 Get and display the list of Marbles

Now we are displaying the list of marbles:

```
marbleRegistry.getAll()
let marbles = await marbleRegistry.getAll();
let tMarbles = new Array({
  head: ['MarbeId', 'Owner', 'Size', 'Color']
});
let arrayLength = tMarbles.length;
marbles.forEach((marble) => {
  let tableLine = [];
  tableLine.push(marble.marbleId);
  tableLine.push(marble.owner);
  tableLine.push(marble.size);
  tableLine.push(marble.color);
  tMarbles.push(tableLine);
})

bizNetConnection.disconnect();

// Put to stdout - as this is really a command line app
return tMarbles;
```

Run this application with the following command:

```
node add-resources.js
```

## 4 Using the composer rest server

In this chapter, we will use the Composer Rest server to publish the transactions as Rest API.

Then we will develop a web application to use these API.

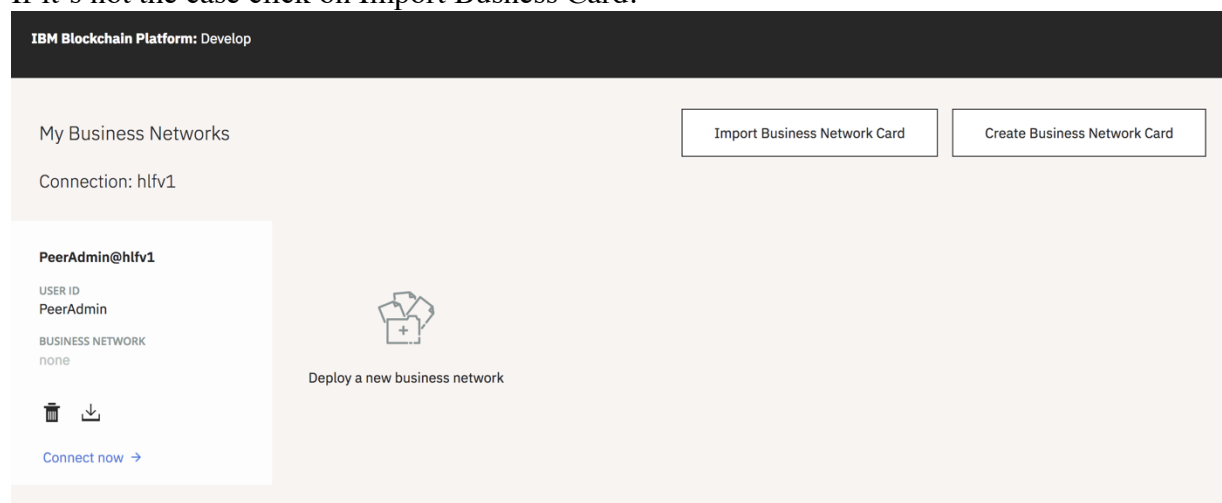
### 4.1 Install the Composer Rest server on IBM Cloud

We will complete the installation done in Lab 1, installing the composer rest server on the blockchain cluster in IBM Cloud.

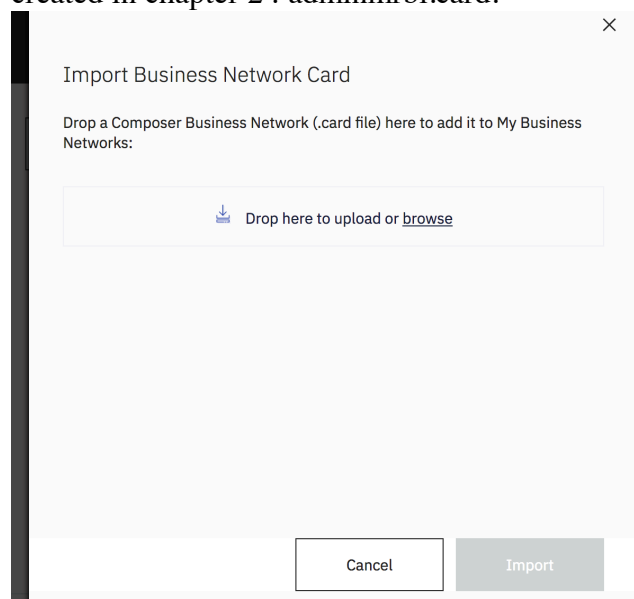
The Business Network card of the admin user of the marble-network will be used to install the Composer Rest Server.

First, check if the business card is imported in the Composer playground environment. You should see it in the Composer Playground view, under “Connection: hlfv1”.

If it's not the case click on Import Business Card.



Then click on Drop here to upload or Browse in order to upload the business network card you have created in chapter 2 : adminmrbl.card.



You should get the following view :

The screenshot shows the 'My Business Networks' section of the IBM Blockchain Platform. It includes two buttons: 'Import Business Network Card' and 'Create Business Network Card'. Below these, there are two network cards. The first card is for 'PeerAdmin@hlfv1' with a 'none' business network. The second card is for 'admin@marbles-network' with a 'marbles-network' business network. Both cards have a 'Connect now' link. To the right, there is a 'Deploy a new business network' button with a plus icon.

Now you are going to create the Rest Server.  
Issue the following commands in a Linux terminal:

```
cd ~/git/hlv11/cs-offerings/scripts
./create/create_composer-rest-server.sh --business-network-card admin@marble-network
```

Check the server, looking at the log in the container :

```
Kubect1 get pods
```

NAME	READY	STATUS	RESTARTS	AGE
blockchain-ca-9cc77f5b9-94khf	1/1	Running	0	1h
blockchain-couchdb1-bf7bbb44-f64cl	1/1	Running	0	1h
blockchain-couchdb2-794dfc7676-t25nw	1/1	Running	0	1h
blockchain-orderer-f4799b879-fztbd	2/2	Running	0	1h
blockchain-org1peer1-6d655c4c68-z4vdv	1/1	Running	0	1h
blockchain-org2peer1-5c589f97b9-lsg6j	1/1	Running	0	1h
composer-playground-7c674c6684-hc8sq	1/1	Running	0	1h
composer-rest-server-689f8bff5b-4ftrr	1/1	Running	0	16m

Display the log issuing the following command

```
kubect1 logs <name of composer-rest-server-xxx>
```

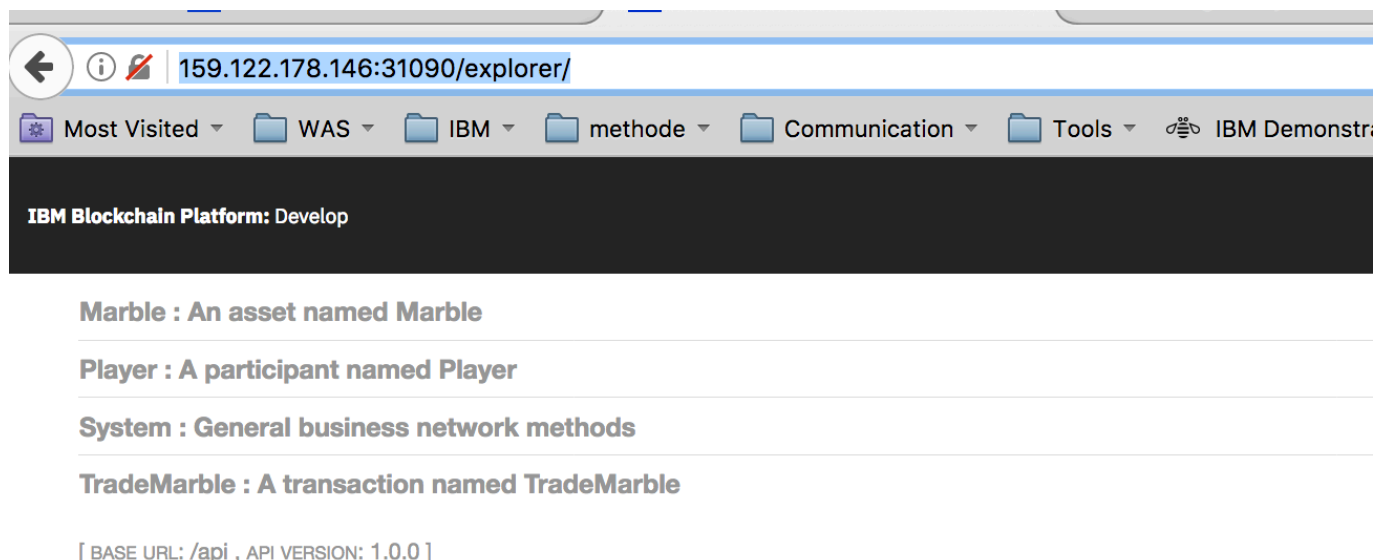
Expected result:

```
WARNING: NODE_APP_INSTANCE value of '0' did not match any instance config file names.
WARNING: See https://github.com/lorenwest/node-config/wiki/Strict-Mode
Discovering types from business network definition ...
Discovered types from business network definition
Generating schemas for all types in business network definition ...
Generated schemas for all types in business network definition
Adding schemas for all types to Loopback ...
Added schemas for all types to Loopback
Web server listening at: http://localhost:3000
Browse your REST API at http://localhost:3000/explorer
```

## 4.2 Check the REST API

To access the Rest explorer, use the port 31090 instead of 3000 with the external ip address.

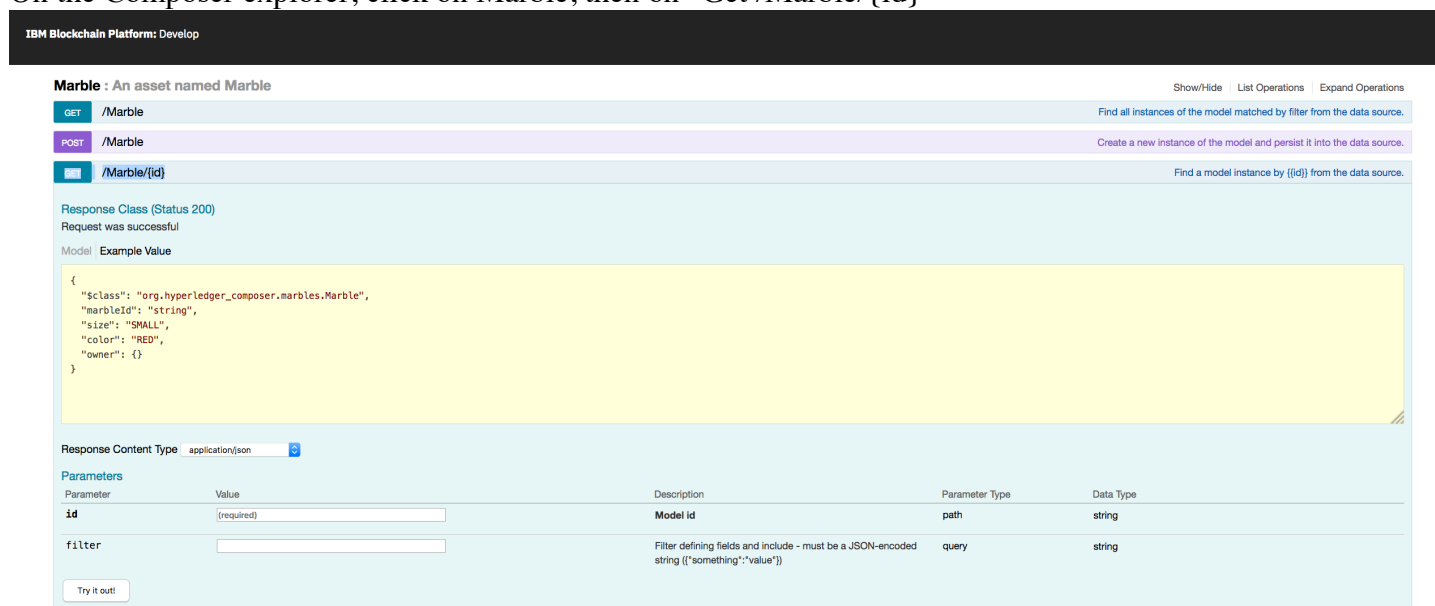
Open your internet browser and access the explorer: [http://<external\\_IP\\_Address>:31090/explorer/](http://<external_IP_Address>:31090/explorer/)



Now you are ready to access to your Business Network through Rest API.  
We are going to explore these API.

#### 4.2.1 Display a marble

On the Composer explorer, click on Marble, then on “Get /Marble/{id}”



In the field id, put “marbleId:1” then click on the “Try it out” button. It will display the content of the asset Marble 1 as shown in the following picture.

**Curl**

```
curl -X GET --header 'Accept: application/json' 'http://159.122.178.146:31090/api/Marble/marbleId%3A1'
```

**Request URL**

```
http://159.122.178.146:31090/api/Marble/marbleId%3A1
```

**Response Body**

```
{
  "$class": "org.hyperledger_composer.marbles.Marble",
  "marbleId": "marbleId:1",
  "size": "MEDIUM",
  "color": "ORANGE",
  "owner": "resource:org.hyperledger_composer.marbles.Player#email:olivier1@mele"
}
```

**Response Code**

```
200
```

**Response Headers**

```
{
  "vary": "Origin, Accept-Encoding",
  "access-control-allow-credentials": "true",
  "x-xss-protection": "1; mode=block",
  "x-frame-options": "DENY",
  "x-download-options": "noopen",
  "x-content-type-options": "nosniff",
  "content-type": "application/json; charset=utf-8",
  "content-length": "188",
  "etag": "W/\"bc-Gv6r+ZDmEgVddhoiWFnzYsaFfeg\"",
  "date": "Sat, 02 Jun 2018 18:54:53 GMT",
  "connection": "keep-alive"
}
```

#### 4.2.2 Execute a transaction

We are going to change the owner of the marbleId:3.

On the Composer explorer, click on TradeMarble, then on “POST /TradeMarble”.

In the field ‘data’, specify the transaction input data:

```
{
  "$class": "org.hyperledger_composer.marbles.TradeMarble",
  "marble": "resource:org.hyperledger_composer.marbles.Marble#marbleId:1",
  "newOwner": "resource:org.hyperledger_composer.marbles.Player#email:olivier4@mele"
}
```

Then click on the button” Try it out” to run the transaction

Parameter	Value	Description	Parameter
data	<pre>{   "\$class": "org.hyperledger_composer.marbles.TradeMarble",   "marble": "resource:org.hyperledger_composer.marbles.Marble#marbleId:3",   "newOwner": "resource:org.hyperledger_composer.marbles.Player#email:olivier4@mele" }</pre>	Model instance data	body

Parameter content type:

[Try it out!](#) [Hide Response](#)

**Curl**

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
  "$class": "org.hyperledger_composer.marbles.TradeMarble", \
  "marble": "resource:org.hyperledger_composer.marbles.Marble#marbleId:3", \
  "newOwner": "resource:org.hyperledger_composer.marbles.Player#email:olivier4@mele" \
}' 'http://159.122.178.146:31090/api/TradeMarble'
```

**Request URL**

```
http://159.122.178.146:31090/api/TradeMarble
```

**Response Body**

```
{
  "$class": "org.hyperledger_composer.marbles.TradeMarble",
  "marble": "resource:org.hyperledger_composer.marbles.Marble#marbleId:3",
  "newOwner": "resource:org.hyperledger_composer.marbles.Player#email:olivier4@mele",
  "transactionId": "1225c508217067fecc2f217f01553b79aa826644f0ea11cd1227ded1ddaa2e84"
}
```

This concludes the lab on composer SDK development. More information on the SDK can be found here:

<https://hyperledger.github.io/composer/jsdoc/index.html>

## 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 MGKer 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](http://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](http://www.ibm.com/legal/copytrade.shtml).



Please Recycle

---