

Oracle BlockChain Cloud Service

Cloud Test Drive Workshop



Photo Credit: DC Chain by Hayden flic.kr/p/dXHir
creativecommons.org/licenses/by/2.0/

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

PaaS – BlockChain Cloud Service (BCS)

Contents:

Demo Attributes	2
Oracle Blockchain Cloud Service.....	2
Overview.....	3
Chapter 0: Recap and starting point	4
Chapter 1: Starting the Blockchain Network.....	8
Chapter 2: Working with the Blockchain – REST API basics	15
Chapter 3: Working with the Blockchain – REST API making changes.....	17
Chapter 4: Working with the Blockchain – Using Postman	19
Chapter 5: Working with the Blockchain – Sample Apps	22

Demo Attributes

Product(s)	Oracle BlockChain Cloud Service
Date last updated	December 2018
Author(s)	Jens Lusebrink
Demo Title(s)	PaaS – BlockChain Cloud Service (OBCS)

Oracle Blockchain Cloud Service

Oracle Blockchain Cloud Service is a new offering that is part of Oracle's comprehensive platform-as-a service (PaaS) portfolio. Delivered by the world's most scalable, distributed transaction processing platform provider, Oracle Blockchain Cloud Service is the most comprehensive distributed ledger cloud platform. A comprehensive distributed ledger cloud platform to provision Blockchain networks, join other organizations, and deploy & run smart contracts to update and query the ledger. Reliably share data and conduct trusted transactions with suppliers, banks, and other trade partners through integration with existing or new cloud-based or on-premises applications.

PaaS – BlockChain Cloud Service (BCS)

Overview

End-to-End Application Flow

This HandsOn Lab showcases the end to end flow of working with an existing Oracle Blockchain Cloud Service (OBCS) network and execute transactions between 3 parties as well as querying the ledger. Both will be shown using the OBSCS REST API and via sample Business Applications.

- Execute transactions on the Blockchain
- Query the Ledger
- Use sample apps to access the Blockchain network

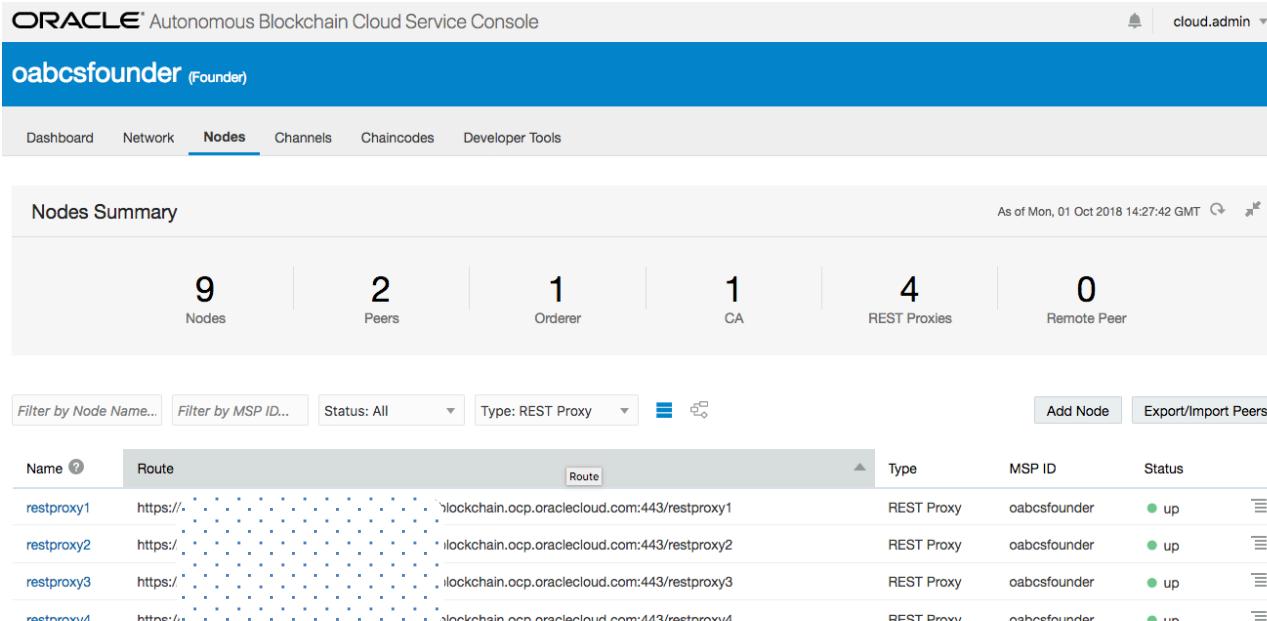
PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
Chapter 0: Recap and starting point		
0.00		<p>Whether you come from the previous lab 'Creating a Blockchain network' or starting fresh with this lab, you'll find a working Blockchain network based on Hyperledger Fabric that will run in the Oracle Cloud on 3 instances (Blockchain Network participants) of Oracle Blockchain Cloud Service (OBSCS). This network simulates the business network between a Car Manufacturer (DetroitAuto, Founder) and 2 Car Dealers (Sam & Jude, Participants). At the end of the previous lab and as a starting point for this lab the Blockchain Network has been populated with some dummy transactions to enable this lab which is all about interacting with the Blockchain Network through the APIs provided by OBSCS.</p> <p>Now, we are going to work on this network and create transactions that will be recorded as blocks into our Blockchain network. The OBSCS can be accessed via the REST API (something that Oracle added on top of HLF) or, in a real-world scenario through business apps interacting via REST with OBSCS.</p> <p>We will begin with the REST API and some sample queries.</p> <p>You can work against the ledger (Query, Invocation) by making REST calls using the terminal or apps like Postman. What you can do against the ledger is defined in the instantiated chaincode.</p>
0.01	ChainCode / Smart Contract	<p>This chapter will provide an overview about the 'logic' of our Blockchain Network. This logic is defined in the ChainCode/SmartContract that is running on the Channels where the Blockchain Network participants are communicating with each other.</p> <p>Chaincode is defining what types of data are written during transactions, providing logic around how transactions are executed and validating the conditions under which they should be run. Chaincode initializes</p>

PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description	
		and manages ledger state through transactions submitted by applications. A chaincode typically handles business logic agreed to by members of the network, so it is also sometimes referred to as a “smart contract”.	
0.02	<p>Looking into the ChainCode we can find the following Methods and Arguments which are defined for our carTrace sample:</p> <p>NOTE: A full list of available Methods and Arguments can be seen in the carTrace.go file on the VM in the /Documents folder.</p>	<p>Method</p> <p>initVehiclePart (add a car part)</p> <p>initVehicle (add a car)</p> <p>transferVehiclePart (transfer ownership)</p> <p>transferVehicle (transfer ownership)</p> <p>readVehiclePart (read records)</p> <p>getHistoryForRecord (read records)</p> <p>deleteVehiclePart (delete records)</p> <p>setPartRecallState (recall a part)</p>	<p>Arguments</p> <p>part id: string assembler: string assembly time: timestamp owner: string recall status: boolean recall time: timestamp</p> <p>vehicle id: string manufacturer: string vehicle name: string assembly time: timestamp airbag serial: string owner: string recall status: Boolean recall time: timestamp</p> <p>part id: string current owner: string new owner: string</p> <p>vehicle id: string current owner: string new owner: string</p> <p>vehicle id: string part id: string</p> <p>part id: string</p> <p>part id: string</p> <p>part id: string status: Boolean</p>

PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description																									
0.03	OBCS REST API	<p>One of the improvements of Oracle Blockchain Cloud Service on top of the Open Source Hyperledger Fabric is the availability of a comprehensive REST API that provides a standard and easy interaction with the chaincode and Blockchain Network and therefore enables a quick integration of business apps with the Blockchain Cloud Service.</p> <p>The REST API is available through a REST proxy on each Blockchain Network participant's instance.</p>  <table border="1"> <thead> <tr> <th>Name</th> <th>Route</th> <th>Type</th> <th>MSP ID</th> <th>Status</th> </tr> </thead> <tbody> <tr> <td>restproxy1</td> <td>https://blockchain.ocp.oraclecloud.com:443/restproxy1</td> <td>REST Proxy</td> <td>oabcsfounder</td> <td>up</td> </tr> <tr> <td>restproxy2</td> <td>https://blockchain.ocp.oraclecloud.com:443/restproxy2</td> <td>REST Proxy</td> <td>oabcsfounder</td> <td>up</td> </tr> <tr> <td>restproxy3</td> <td>https://blockchain.ocp.oraclecloud.com:443/restproxy3</td> <td>REST Proxy</td> <td>oabcsfounder</td> <td>up</td> </tr> <tr> <td>restproxy4</td> <td>https://blockchain.ocp.oraclecloud.com:443/restproxy4</td> <td>REST Proxy</td> <td>oabcsfounder</td> <td>up</td> </tr> </tbody> </table> <p>You can use the terminal window or any preferred REST tooling like Postman calling the gateway. More info can be found in the documentation: https://docs.oracle.com/en/cloud/paas/blockchain-cloud/user/develop-blockchain-applications.html</p>	Name	Route	Type	MSP ID	Status	restproxy1	https://blockchain.ocp.oraclecloud.com:443/restproxy1	REST Proxy	oabcsfounder	up	restproxy2	https://blockchain.ocp.oraclecloud.com:443/restproxy2	REST Proxy	oabcsfounder	up	restproxy3	https://blockchain.ocp.oraclecloud.com:443/restproxy3	REST Proxy	oabcsfounder	up	restproxy4	https://blockchain.ocp.oraclecloud.com:443/restproxy4	REST Proxy	oabcsfounder	up
Name	Route	Type	MSP ID	Status																							
restproxy1	https://blockchain.ocp.oraclecloud.com:443/restproxy1	REST Proxy	oabcsfounder	up																							
restproxy2	https://blockchain.ocp.oraclecloud.com:443/restproxy2	REST Proxy	oabcsfounder	up																							
restproxy3	https://blockchain.ocp.oraclecloud.com:443/restproxy3	REST Proxy	oabcsfounder	up																							
restproxy4	https://blockchain.ocp.oraclecloud.com:443/restproxy4	REST Proxy	oabcsfounder	up																							

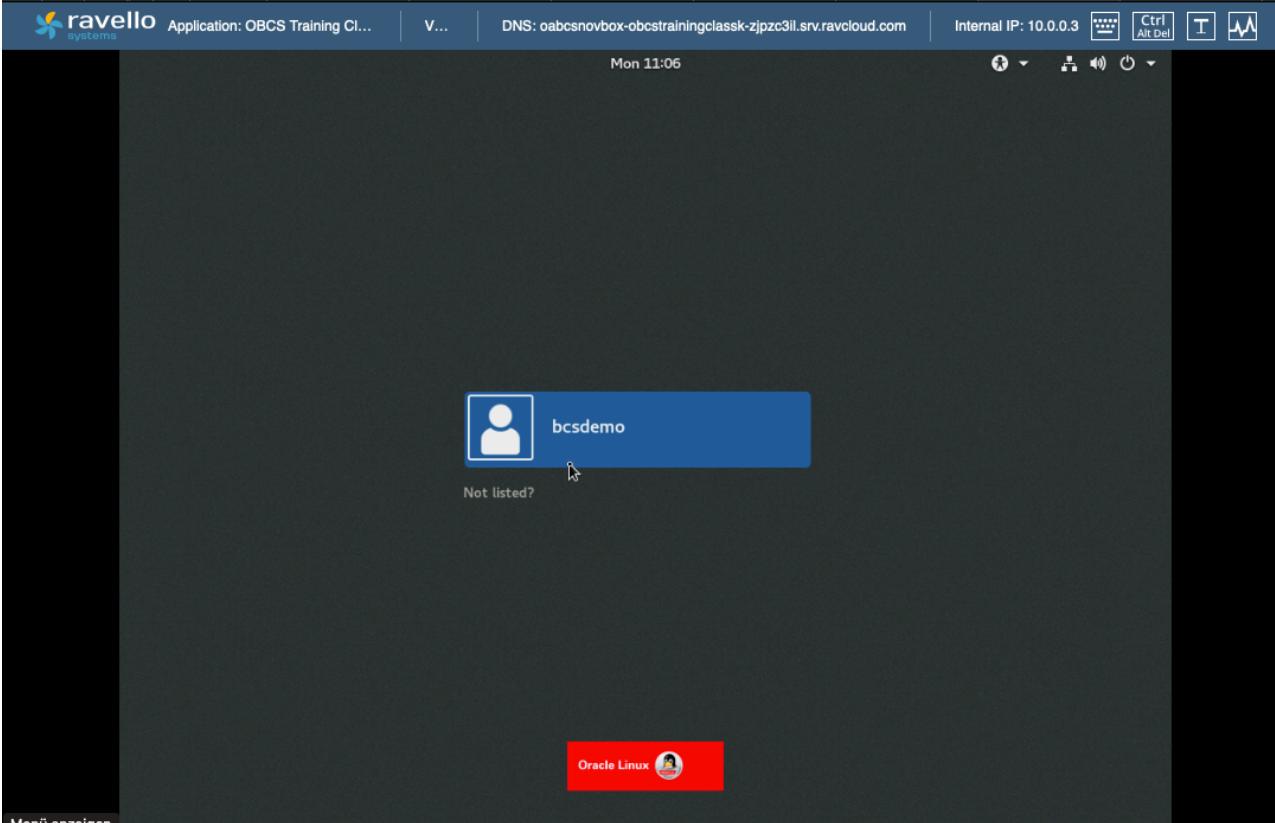
PaaS – BlockChain Cloud Service (BCS)

0.04	Blockchain transactions	<p>During the creation of the Blockchain Network, the ledger has been populated with some sample transaction. You can see the key/value pairs in the table:</p> <table><tbody><tr><td>Channel</td><td>PartID</td></tr><tr><td>Samchannel</td><td>abg1234, abg1235, ser1236, win1237, bra1238</td></tr><tr><td>Judechannel</td><td>abg1239, abg1240, whl1241, win1242, sen1243</td></tr><tr><td>Channel</td><td>vehicleID</td></tr><tr><td>Samchannel</td><td>dtrt10001, dtrt10002</td></tr></tbody></table>	Channel	PartID	Samchannel	abg1234, abg1235, ser1236, win1237, bra1238	Judechannel	abg1239, abg1240, whl1241, win1242, sen1243	Channel	vehicleID	Samchannel	dtrt10001, dtrt10002
Channel	PartID											
Samchannel	abg1234, abg1235, ser1236, win1237, bra1238											
Judechannel	abg1239, abg1240, whl1241, win1242, sen1243											
Channel	vehicleID											
Samchannel	dtrt10001, dtrt10002											
<p><u>Congratulation</u> <u>You completed Chapter 1</u></p>												

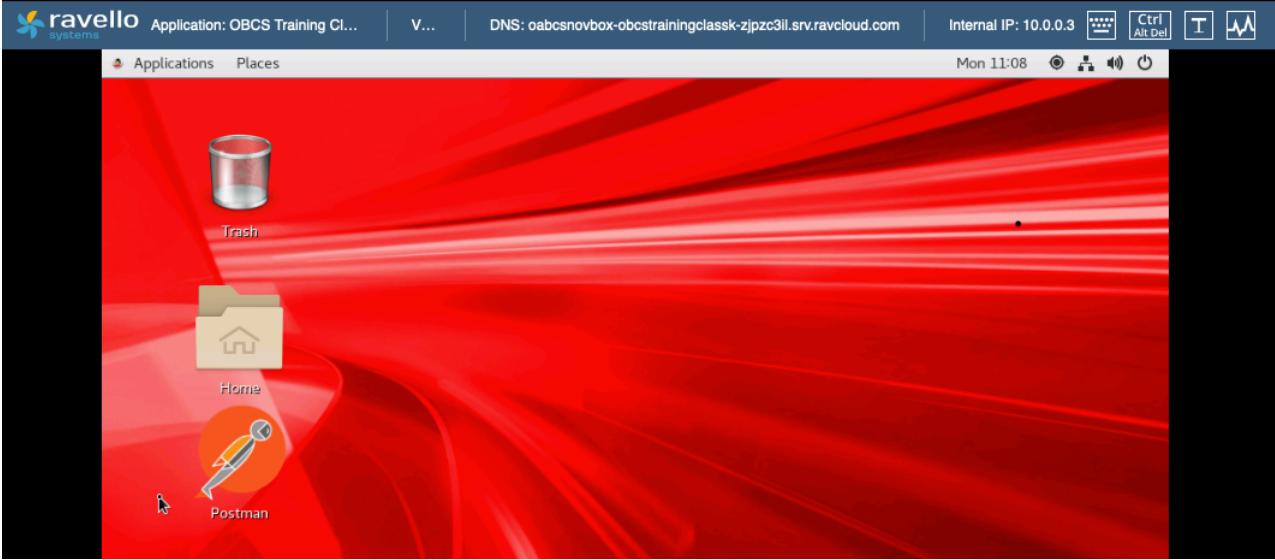
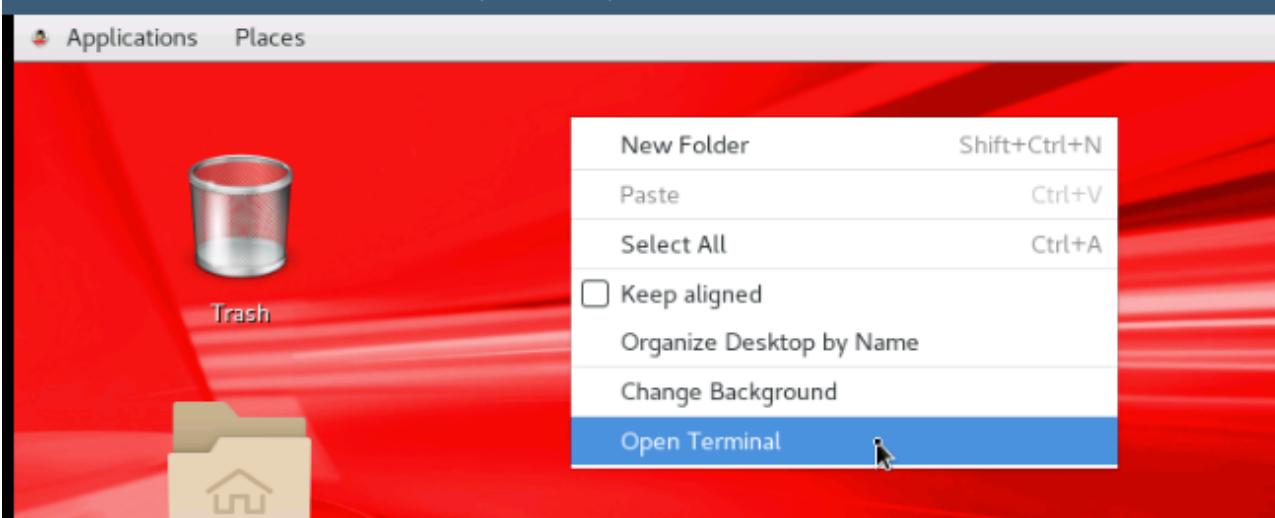
PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
Chapter 1: Starting the Blockchain Network		
1.00		For the sake of this hands-on-lab, we need to start our Blockchain network which consists of the abovementioned 3 participants. In the real world this is not necessary.
1.01	1. Go to the VM console window 2. Click on 'Console'	<p>⌚ Stopping in: 27:29 (hr:min)</p> <p>Started</p> <p>oabcs_novbox</p> <p>SERVICES MORE ▾</p> <p>HTTPS</p> <p>HTTP: 130.61.66.110 Port: 80,2000-4000,7000-9000</p> <p>CONSOLE</p>

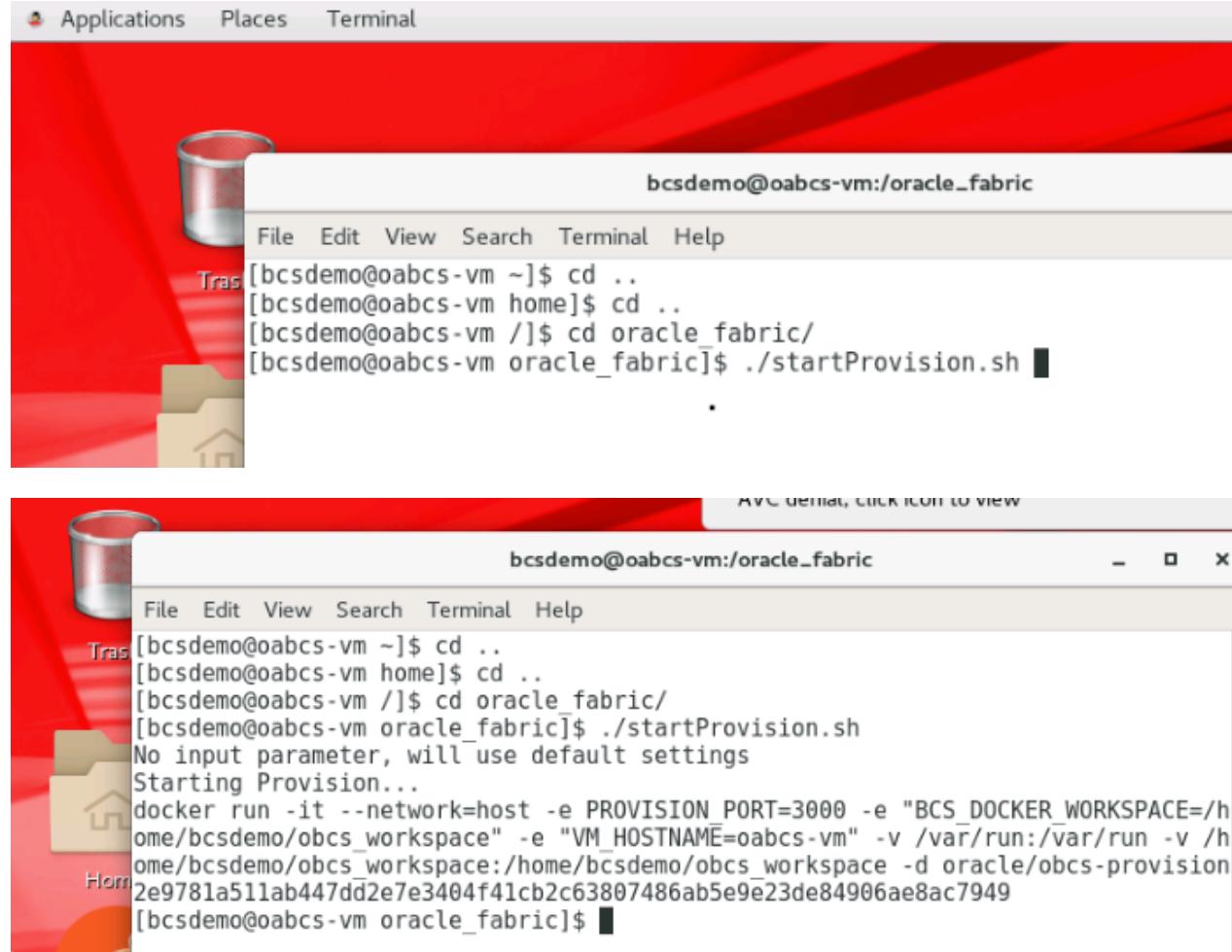
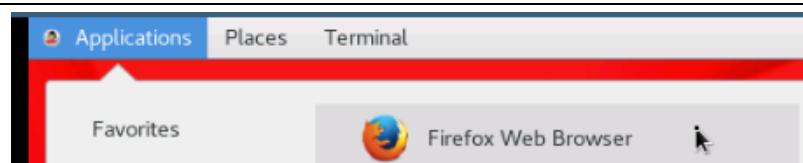
PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
1.02	3. A new window will appear with the VM desktop	 A screenshot of a Ravello Systems application window titled "ravello systems Application: OBCS Training Cl...". The window shows a dark Linux desktop environment. At the top, there is a toolbar with the Ravello logo, application name, and various icons. The main area displays a user icon labeled "bcsdemo" with the status "Not listed?". A cursor arrow is visible near the bottom right of the desktop. The bottom of the window features a red footer bar with the text "Oracle Linux" and a small icon.

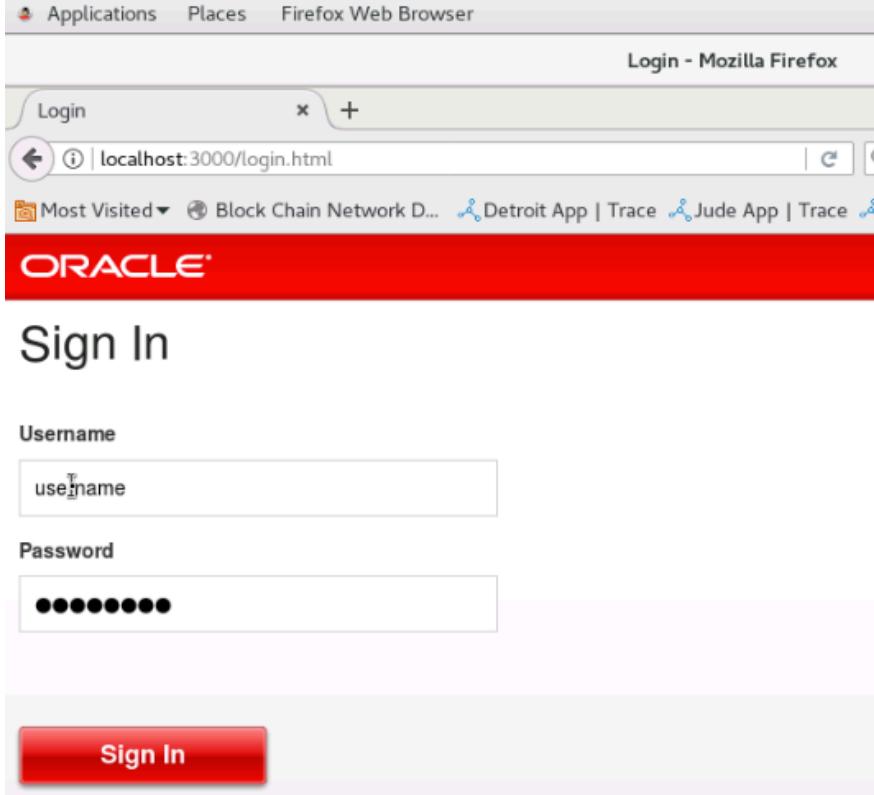
PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
1.03	4. Hit enter to access the desktop	
1.04	5. On the desktop, right-mouse click and select 'Open in Terminal'	

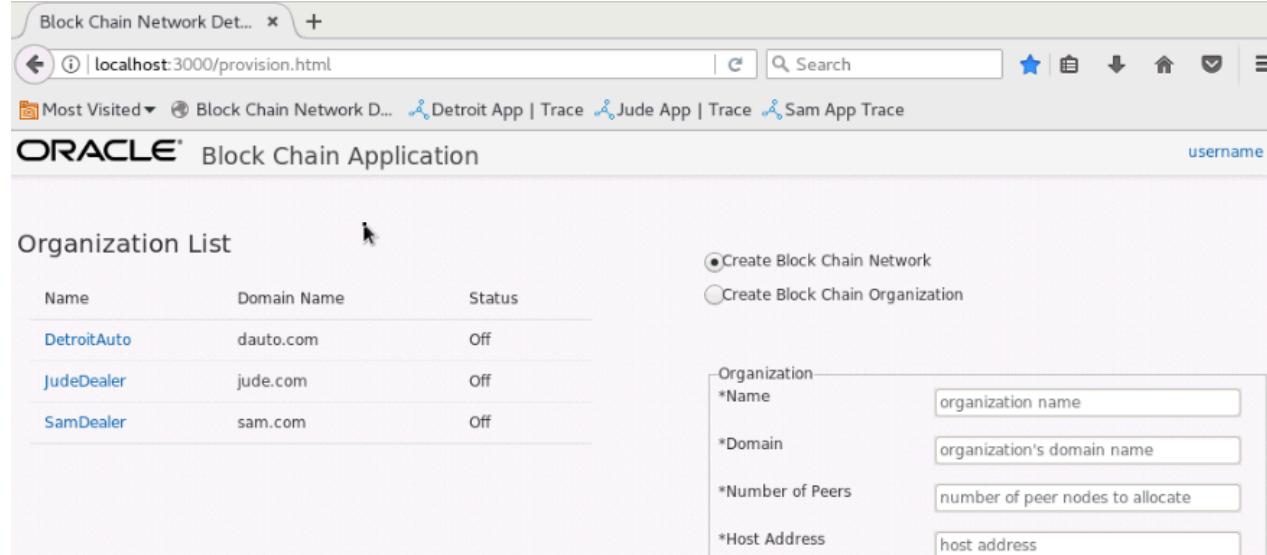
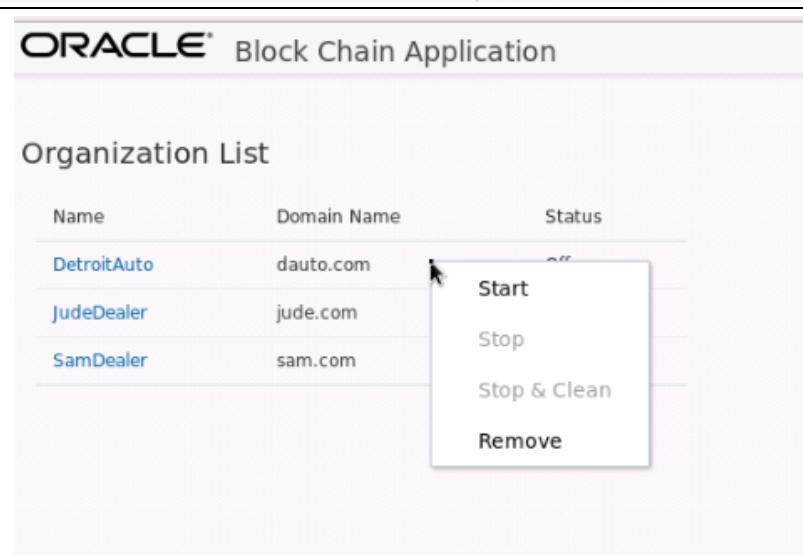
PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
1.05	<p>6. In the terminal window, enter the following commands</p> <ul style="list-style-type: none"> a. cd .. b. cd .. c. cd oracle_fabric d. ./startProvision.sh <p>NOTE: Close each command by pressing 'ENTER' on your keyboard</p> <p>7. The Oracle Blockchain Cloud Service will start</p>	 <pre> [bcsmo@oabcs-vm ~]\$ cd .. [bcsmo@oabcs-vm home]\$ cd .. [bcsmo@oabcs-vm /]\$ cd oracle_fabric/ [bcsmo@oabcs-vm oracle_fabric]\$./startProvision.sh [bcsmo@oabcs-vm oracle_fabric]\$ [bcsmo@oabcs-vm ~]\$ cd .. [bcsmo@oabcs-vm home]\$ cd .. [bcsmo@oabcs-vm /]\$ cd oracle_fabric/ [bcsmo@oabcs-vm oracle_fabric]\$./startProvision.sh No input parameter, will use default settings Starting Provision... docker run -it --network=host -e PROVISION_PORT=3000 -e "BCS_DOCKER_WORKSPACE=/home/bcsmo/oabcs_workspace" -e "VM_HOSTNAME=oabcs-vm" -v /var/run:/var/run -v /home/bcsmo/oabcs_workspace:/home/bcsmo/oabcs_workspace -d oracle/oabcs-provision 2e9781a511ab447dd2e7e3404f41cb2c63807486ab5e9e23de84906ae8ac7949 [bcsmo@oabcs-vm oracle_fabric]\$ </pre>
1.06	8. From the 'Applications' menu, select Firefox web browser	

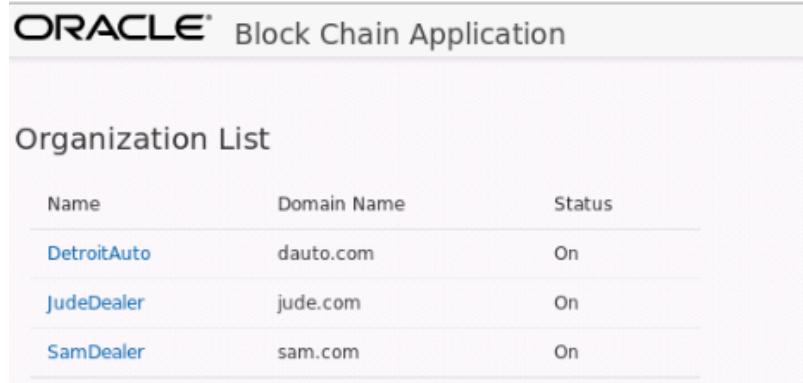
PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
	<p>9. In Firefox, enter 'localhost:3000' in the address bar.</p> <p>10. In the logon screen enter Username = username Password = password</p> <p>11. Click on 'Sign In'</p>	 <p>The screenshot displays a Firefox browser window with the title "Login - Mozilla Firefox". The address bar contains "localhost:3000/login.html". Below the address bar, the page header features the "ORACLE" logo. The main content area is titled "Sign In". It includes two input fields: "Username" with the value "username" and "Password" with several dots indicating the password. A large red "Sign In" button is positioned at the bottom of the form.</p>

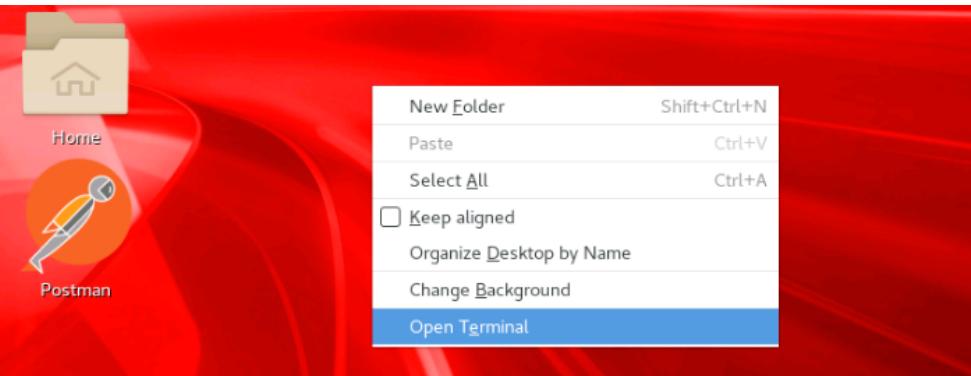
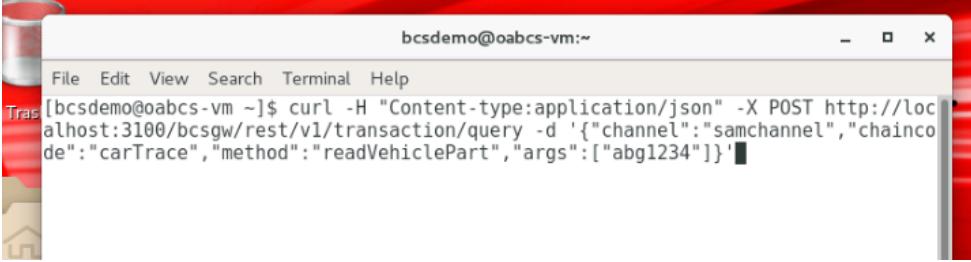
PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
1.07	12. The Management UI starts and you can see the existing network. 13. The network is currently turned OFF	 A screenshot of a web browser window titled "Block Chain Network Det...". The address bar shows "localhost:3000/provision.html". The page header says "ORACLE® Block Chain Application". On the left, there's a table titled "Organization List" with three rows: DetroitAuto (dauto.com, Off), JudeDealer (jude.com, Off), and SamDealer (sam.com, Off). On the right, there are two radio buttons: "Create Block Chain Network" (selected) and "Create Block Chain Organization". Below them is a form with fields: "Organization", "*Name" (text input "organization name"), "*Domain" (text input "organization's domain name"), "*Number of Peers" (text input "number of peer nodes to allocate"), and "*Host Address" (text input "host address").
1.08	14. Turn ON the network by right-mouse click on the line for each network member and select 'Start'. 15. Refresh your web page to see if all instances are started.	 A screenshot of the same management interface. The "Organization List" table shows the same three entries. A context menu is open over the "SamDealer" row, listing options: "Start", "Stop", "Stop & Clean", and "Remove".

PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description												
	16. You should see all member are ON.	 <p>The screenshot shows the Oracle Block Chain Application interface. At the top, it says "ORACLE® Block Chain Application". Below that is a section titled "Organization List". A table lists three organizations:</p> <table><thead><tr><th>Name</th><th>Domain Name</th><th>Status</th></tr></thead><tbody><tr><td>DetroitAuto</td><td>dauto.com</td><td>On</td></tr><tr><td>JudeDealer</td><td>jude.com</td><td>On</td></tr><tr><td>SamDealer</td><td>sam.com</td><td>On</td></tr></tbody></table>	Name	Domain Name	Status	DetroitAuto	dauto.com	On	JudeDealer	jude.com	On	SamDealer	sam.com	On
Name	Domain Name	Status												
DetroitAuto	dauto.com	On												
JudeDealer	jude.com	On												
SamDealer	sam.com	On												

PaaS – BlockChain Cloud Service (BCS)

Chapter 2: Working with the Blockchain – REST API basics	
2.00	<p>In this chapter we will begin with the REST API and try some sample queries against our Blockchain Network. You can work against the ledger (Query, Invocation) by making REST calls using the terminal or apps like Postman. What you can do against the ledger is defined in the instantiated chaincode (see previous chapter).</p>
2.01	<p>The default syntax for working on the Command Line looks like the following:</p> <p>NOTE: The method and arguments depend on your choice of method. See previous section.</p> <p>You could either do an Invocation or Query.</p> <pre>Curl -H "Content-type:application/json" -X POST http://yourRESTProxy_endpoint:PORT/bcsgw/rest/v1/transaction/invocation -d { "channel":"<channel_name>", "chaincode":"<chaincode_name>", "method":"<function_name>", "args":<arguments as an array>, "chaincodeVer":"<chaincode_version>" }</pre>
2.02	<p>Let's do a query against the ledger for a specific car part.</p> <ol style="list-style-type: none"> 1. Open a terminal window  <ol style="list-style-type: none"> 2. Enter the following sequence at the cursor position. 

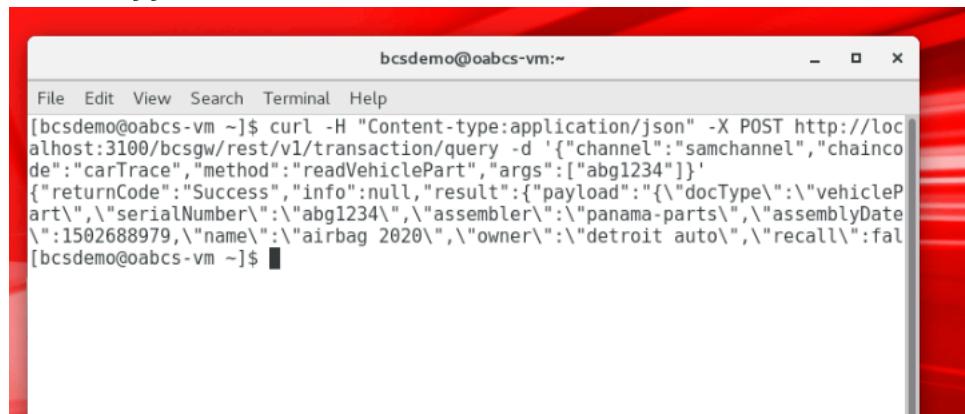
PaaS – BlockChain Cloud Service (BCS)

3. Hit 'Return' to submit the command.

```
curl -H "Content-type:application/json" -X POST http://localhost:3100/bcsgw/rest/v1/transaction/query -d '{"channel":"samchannel","chaincode":"carTrace","method":"readVehiclePart","args":["abg1234"]}'
```

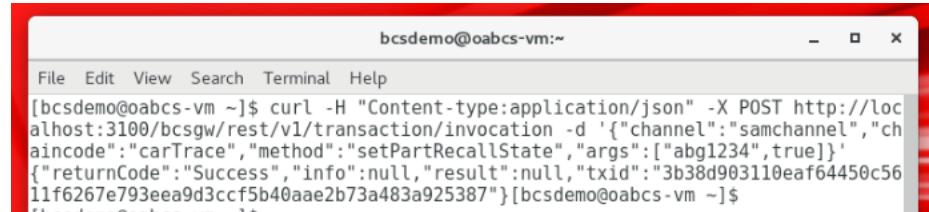
4. The expected outcome should look like this (formatted for better readability).

```
{"returnCode":"Success","info":null,"result":{  
"payload":  
"{"\\"docType\\":\\"vehiclePart\\",  
"\\"serialNumber\\":\\"abg1234\\",  
"\\"assembler\\":\\"panama-parts\\",  
"\\"assemblyDate\\":1502688979,  
"\\"name\\":\\"airbag 2020\\",  
"\\"owner\\":\\"detroit auto\\",  
"\\"recall\\":false,  
"\\"recallDate\\":1502688979}",  
"encode":"UTF-8"}}
```

**Congratulation**

You completed Chapter 2 and have accessed the Blockchain Ledger from a Terminal window.

PaaS – BlockChain Cloud Service (BCS)

Chapter 3: Working with the Blockchain – REST API making changes		
3.00	<p>Now we want to make use of a feature in our Chaincode / SmartContract.</p> <p>Assuming that the Car manufacturer would like to recall a specific part delivered to the dealer. This part (record) must be changed and marked as recalled. As we cannot change the Blockchain, we have to add a new block where we append the existing information with the recall.</p>	
3.01	<p>We are going to “invoke” a method on the “carTrace” chaincode that will effectively mark a particular part as recalled via a REST call.</p> <ol style="list-style-type: none"> 1. Type the following sequence into your terminal window. 2. Hit ‘Return’ to submit the command. 	<pre>curl -H "Content-type:application/json" -X POST http://127.0.0.1:3100/bcsgw/rest/v1/transaction/invocation -d '{"channel":"samchannel","chaincode":"carTrace","method":"setPartRecallState","args":["abg1234",true]}'</pre>
3.02	<p>3. The expected outcome should look like this</p> <p>NOTE: The REST API is giving back the result of the command. As we have done an Invocation instead of a Query, we are just getting ‘Success’ back.</p>	<pre>{"returnCode":"Success","info":null,"result":null,"txid":"583f3ca55b9f2860875b8978e386611b476892c348440a248f07e98dbdad419d"}</pre>  <pre>bcsdemo@oabcs-vm:~ - □ × File Edit View Search Terminal Help [bcsdemo@oabcs-vm ~]\$ curl -H "Content-type:application/json" -X POST http://localhost:3100/bcsgw/rest/v1/transaction/invocation -d '{"channel":"samchannel","chaincode":"carTrace","method":"setPartRecallState","args":["abg1234",true]}' {"returnCode":"Success","info":null,"result":null,"txid":"583f3ca55b9f2860875b8978e386611b476892c348440a248f07e98dbdad419d"}[bcsdemo@oabcs-vm ~]\$</pre>
3.03	In order to verify the change (recall), let's execute the query from Chapter 2 again:	<pre>curl -H "Content-type:application/json" -X POST http://localhost:3100/bcsgw/rest/v1/transaction/query -d '{"channel":"samchannel","chaincode":"carTrace","method":"readVehiclePart","args":["abg1234"]}'</pre>
3.04	This is the result of this query:	<pre>{"returnCode":"Success","info":null,"result": {"payload": "\\"docType\\":\\"vehiclePart\\", \\"serialNumber\\":\\"abg1234\\", \\"assembler\\":\\"panama-parts\\", \\"assemblyDate\\":1502688979, \\"name\\":\\"airbag 2020\\", \\"owner\\":\\"detroit auto\\", \\"recall\\":true,</pre>

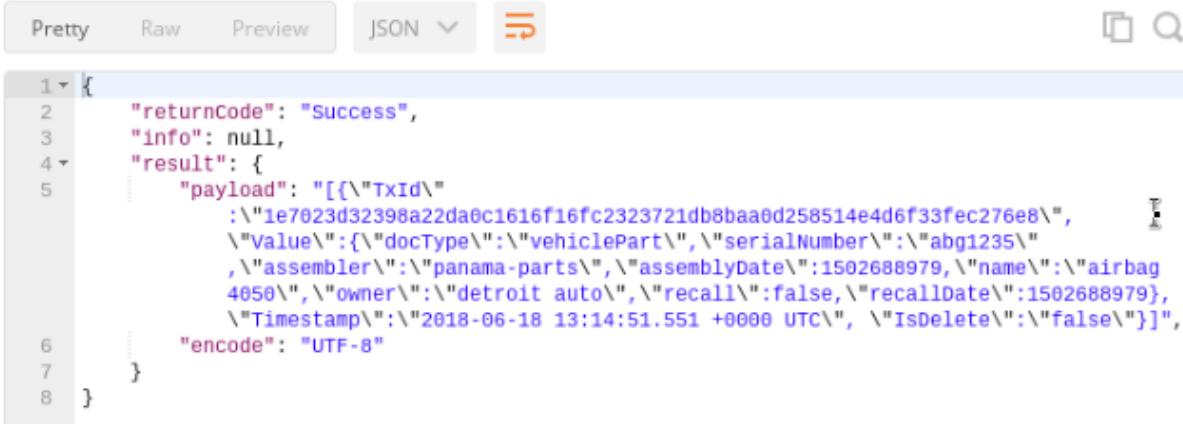
PaaS – BlockChain Cloud Service (BCS)

		<pre>\\"recallDate\":1502688979}", "encode\":\"UTF-8\"}}</pre>
3.05		Each transaction in the Blockchain will be appended to the end of the previous transaction forming a chain of blocks. If we would like to see changes associated to an item we have to create a query against the transaction history.
3.06	1. In the terminal window enter the following sequence	curl -H "Content-type:application/json" -X POST http://localhost:3100/bcsgw/rest/v1/transaction/ query -d '{"channel":"samchannel","chaincode":"carTrace","method":" getHistoryForRecord ","args":["abg1234"]}'
3.07	2. This time the result is more complex as it contains all transaction details for the specific partID.	
Congratulations		
In this chapter you have learned how to interact with the Blockchain Cloud Service through the REST API using the defined methods from the ChainCode/SmartContract.		

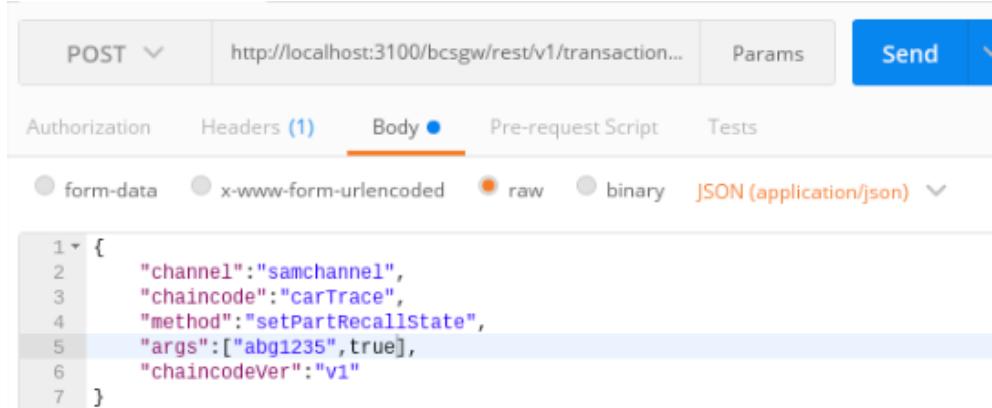
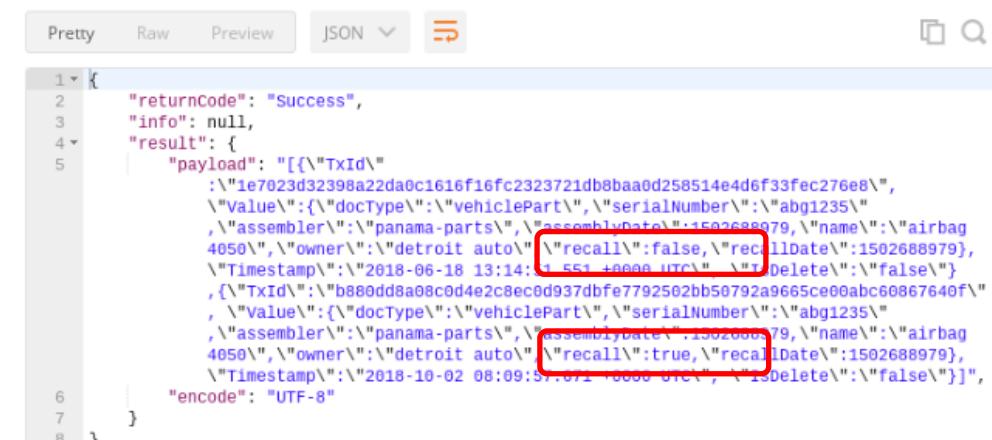
PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
Chapter 4: Working with the Blockchain – Using Postman		
4.00	<p>Another way to interact with the Blockchain Cloud Service is to use a tool that is using the REST API to connect to the service and provides a User Interface. Such a tool is e.g. Postman (https://www.getpostman.com/). We are using Postman to query the Blockchain using the REST API.</p> <ol style="list-style-type: none"> 1. Launch Postman from the VMs Desktop: 	
4.01	<p>Query some details from the Blockchain Network. We will check if a particular record has been recorded at the Blockchain.</p> <ol style="list-style-type: none"> 2. In Postman, enter the following details: <p>Note: We are using again the method 'getHistory For Record' in this example.</p> 3. Click 'Send' 	<p>Modus: POST URL: http://localhost:3100/bcsgw/rest/v1/transaction/query</p> <p>Headers: <input checked="" type="checkbox"/> Content-Type application/json</p> <p>Body – select 'RAW' and paste:</p> <pre>{ "channel": "samchannel", "chaincode": "carTrace", "method": "getHistoryForRecord", "args": ["abg1235"], "chaincodeVer": "v1" }</pre> <p>POST ▾ http://localhost:3100/bcsgw/rest/v1/transaction/query Params Send ▾</p> <p>Authorization Headers (1) Body ● Pre-request Script Tests</p> <p>form-data x-www-form-urlencoded raw binary JSON (application/json) ▾</p> <pre>1 { 2 "channel": "samchannel", 3 "chaincode": "carTrace", 4 "method": "getHistoryForRecord", 5 "args": ["abg1235"], 6 "chaincodeVer": "v1" 7 }</pre>

PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
4.02	In the query result, we can see that the record (abg1235) exists in the Blockchain with the transactionID and the transaction details. All this is info that has been submitted to the record when the ChainCode / Smart Contract has been executed.	 <pre> 1 { 2 "returnCode": "Success", 3 "info": null, 4 "result": { 5 "payload": "[{\\"TxId\\" 6 : \"1e7023d32398a22da0c1616f16fc2323721db8baa0d258514e4d6f33fec276e8\", 7 \"Value\":{\"docType\":\"vehiclePart\", \"serialNumber\":\"abg1235\", 8 \"assembler\":\"panama-parts\", \"assemblyDate\":1502688979, \"name\":\"airbag 4050\", \"owner\":\"detroit auto\", \"recall\":false, \"recallDate\":1502688979}, 9 \"Timestamp\":\"2018-06-18 13:14:51.551 +0000 UTC\", \"IsDelete\":\"false\"}]", 10 "encode": "UTF-8" 11 } 12 } </pre>
4.03	Of course, we can submit the same commands as we did in the previous chapter. E.g. if we want to set the part as recalled, we can enter the following lines in Postman:	<p>Modus: POST URL: <code>http://yourRESTProxy_endpoint:PORT/bcsgw/rest/v1/transaction/invocation</code></p> <p>Headers: <input checked="" type="checkbox"/> Content-Type application/json</p> <p>Body – select 'RAW' and paste:</p> <pre>{ "channel":"samchannel", "chaincode":"carTrace", "method":"setPartRecallState", "args":["abg1235",true], "chaincodeVer":"v1" }</pre>

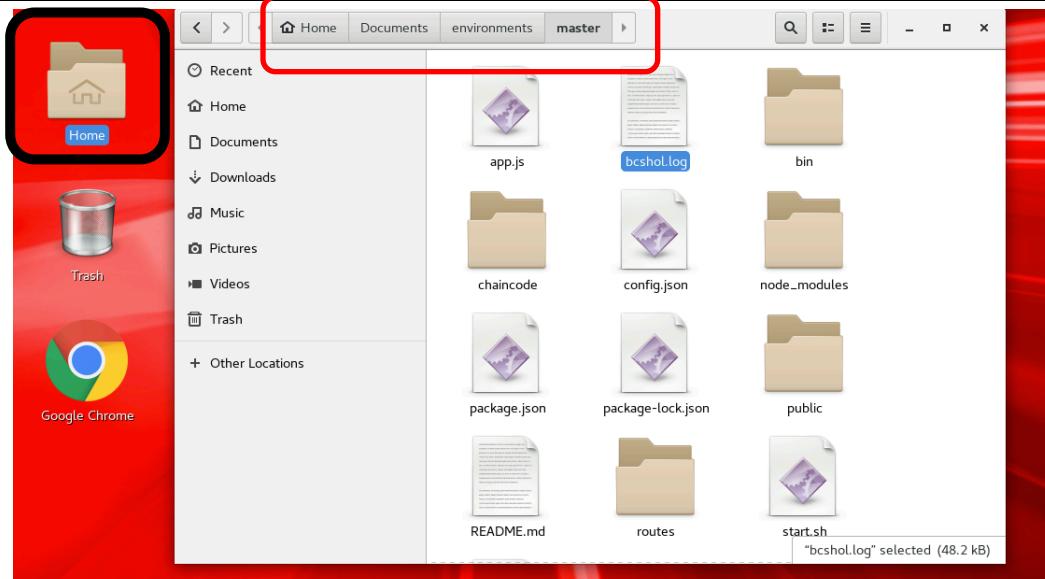
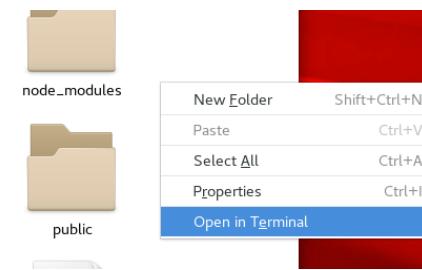
PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
		 <pre> 1 { 2 "channel": "samchannel", 3 "chaincode": "carTrace", 4 "method": "setPartRecallstate", 5 "args": ["abg1235", true], 6 "chaincodeVer": "v1" 7 }</pre>
4.04	And the expected outcome in Postman.	 <pre> 1 { 2 "returnCode": "Success", 3 "info": null, 4 "result": null, 5 "txid": "b880dd8a08c0d4e2c8ec0d937dbfe7792502bb50792a9665ce00abc60867640f" 6 }</pre>
4.05	To verify that the information has been updated we will run the 'getHistoryFor Record' again. See the result in Postman.	 <pre> 1 { 2 "returnCode": "Success", 3 "info": null, 4 "result": { 5 "payload": "[{\\"TxId\\" 6 : "\\ie7023d32398a22da0c1616f16fc2323721db8baa0d258514e4d6f33fec276e8\\", 7 "\\Value\\": {"docType": "vehiclePart", "serialNumber": "abg1235", 8 , "assembler": "panama-parts", "assemblyDate": "1502688979", "name": "airbag 4050", "owner": "detroit auto", "recall": false, "recalibrate": "1502688979"}, 9 , "TxId": "b880dd8a08c0d4e2c8ec0d937dbfe7792502bb50792a9665ce00abc60867640f", 10 , "Value": {"docType": "vehiclePart", "serialNumber": "abg1235", 11 , "assembler": "panama-parts", "assemblyDate": "1502688979", "name": "airbag 4050", "owner": "detroit auto", "recall": true, "recalibrate": "1502688979", 12 , "Timestamp": "2018-06-18 13:14:55 +0000 UTC", "isDelete": "false"}] 13 } 14 } 15 }</pre>

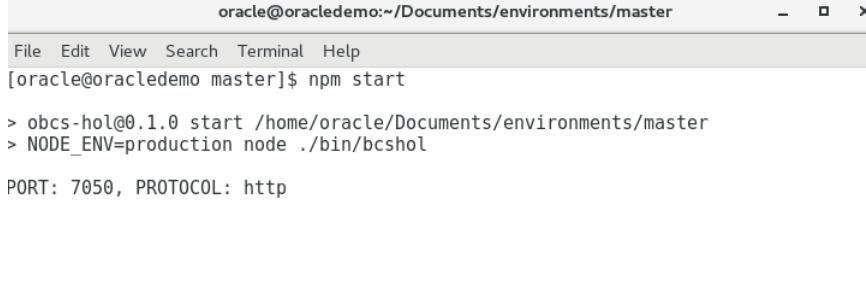
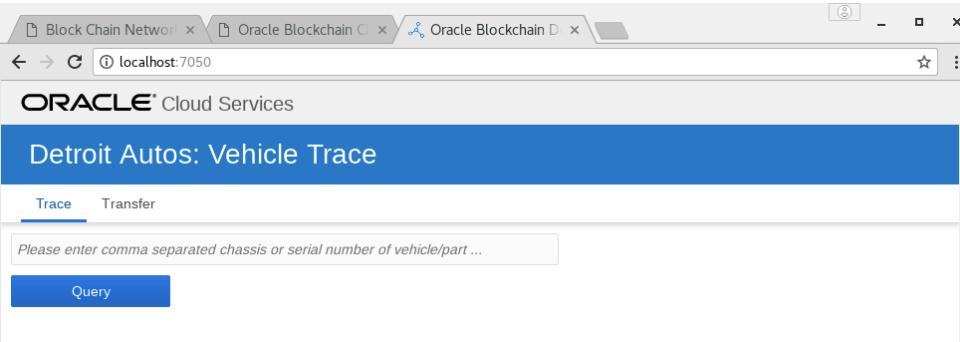
Congratulations

You have just learned a different way to interact with the Blockchain Cloud Service using Postman and the REST API

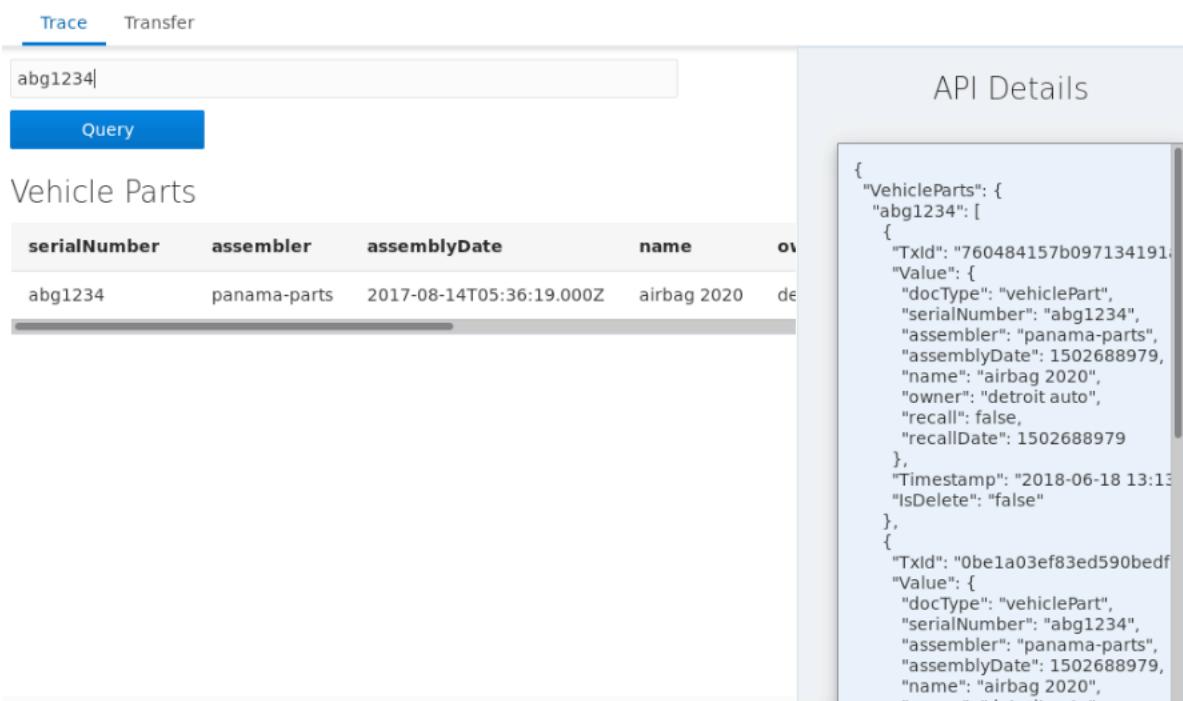
PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
Chapter 5: Working with the Blockchain – Sample Apps		
5.00		<p>In a real business scenario, Blockchain will be embedded in Business Applications that trigger transactions on the Blockchain. For demo purpose we have built a sample app that mimics such behaviour and let you work with the Oracle Blockchain Cloud Service.</p>
5.01	<ol style="list-style-type: none"> On your Blockchain VM open the 'Home' directory and select 'Documents/environments' 	 <ol style="list-style-type: none"> Open the 'master' directory and right-mouse click in the background Select 'Open Terminal' from the menu. 

PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
	<p>4. In the Terminal window enter 'npm start'. 5. The sample app will start.</p> <p>Note: You can monitor the process in the terminal window.</p>	 <pre>oracle@oracledemo:~/Documents/environments/master File Edit View Search Terminal Help [oracle@oracledemo master]\$ npm start > obcs-hol@0.1.0 start /home/oracle/Documents/environments/master > NODE_ENV=production node ./bin/bcshol PORT: 7050, PROTOCOL: http</pre>
5.02	<p>1. Open the webbrowser and 2. type in the following URL: http://localhost:7050 3. This will open the sample app UI for the Car manufacturer.</p>	 <p>The screenshot shows a web browser window titled "Detroit Autos: Vehicle Trace". The address bar says "localhost:7050". The page header reads "ORACLE® Cloud Services". Below it, there are two tabs: "Trace" (which is selected) and "Transfer". A text input field contains the placeholder "Please enter comma separated chassis or serial number of vehicle/part ...". A blue "Query" button is at the bottom of the input field.</p>
5.03	<p>The sample application includes simple web interfaces interacting directly with OBCS using RESTful API's. Typically, in a real-world setup, you could imagine a scenario where the supply chain application on each end directly integrates with OBCS using REST. The dealers push transactions to a private channel shared with the manufacturer e.g. "samchannel", executed by a smart contract called "carTrace" which is currently in v1 of its release. This smart contract is responsible for all state transitions on the consortium's permissioned ledger.</p>	

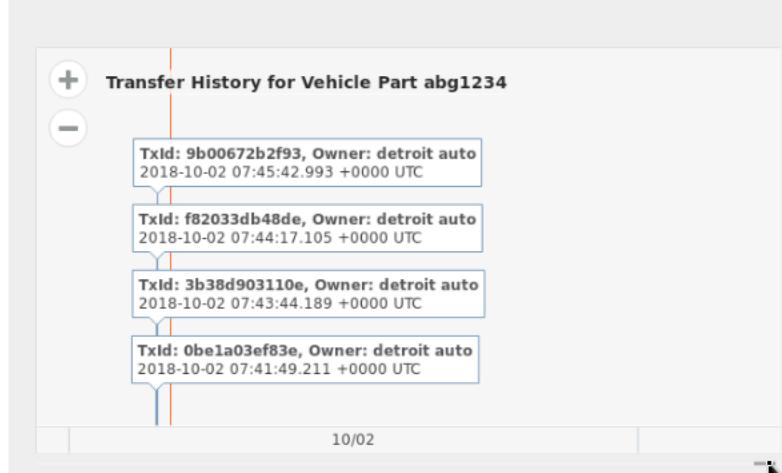
PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
5.04	<p>Working with the Blockchain. Querying the ledger for automotive parts.</p> <ol style="list-style-type: none"> Enter some valid part numbers in the Query field <p>Valid part numbers: abg1234 abg1235</p> <ol style="list-style-type: none"> You can see the part details alongside with the REST API result set. In our sample app the query feature always retrieves the full history of the partID using the 'getHistoryForRecord' method. <p>Again, the terminal window provides the raw output.</p>	 <pre>{ "VehicleParts": { "abg1234": [{ "TxId": "760484157b097134191", "Value": { "docType": "vehiclePart", "serialNumber": "abg1234", "assembler": "panama-parts", "assemblyDate": 1502688979, "name": "airbag 2020", "owner": "detroit auto", "recall": false, "recallDate": 1502688979 }, "Timestamp": "2018-06-18 13:13:13", "IsDelete": "false" }, { "TxId": "Obela03ef83ed590bedf", "Value": { "docType": "vehiclePart", "serialNumber": "abg1234", "assembler": "panama-parts", "assemblyDate": 1502688979, "name": "airbag 2020", "owner": "detroit auto" } }] } }</pre>

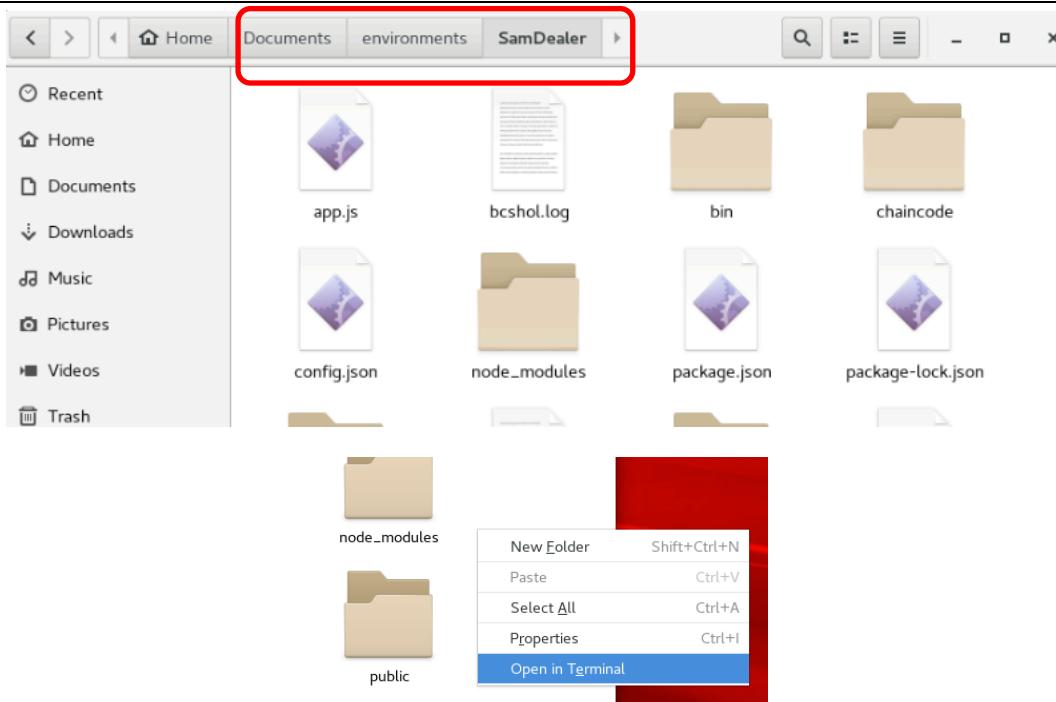
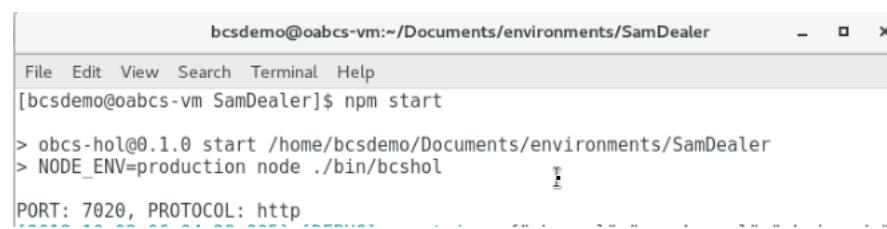
PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
		<pre> File Edit View Search Terminal Help > obcs-hol@0.1.0 start /home/oracle/Documents/environments/master > NODE_ENV=production node ./bin/bcshol PORT: 7050, PROTOCOL: http [2018-02-27 06:33:15.581] [DEBUG] agent.js - {"channel":"sam.channel","chaincode":"carTrace", "method":"getHistoryForRecord","args":["abg1234"],"chaincodeVer":"v1"} [2018-02-27 06:33:15.584] [DEBUG] agent.js - {"channel":"jude.channel","chaincode":"carTrace", "method":"getHistoryForRecord","args":["abg1234"],"chaincodeVer":"v1"} [2018-02-27 06:33:15.738] [DEBUG] agent.js - STATUS: 200 [2018-02-27 06:33:15.738] [DEBUG] agent.js - HEADERS: {"content-type": "application/json", "con- tent-length": "50", "date": "Tue, 27 Feb 2018 11:33:15 GMT", "connection": "close"} [2018-02-27 06:33:15.739] [DEBUG] agent.js - BODY: {"returnCode": "Success", "result": "[]", "inf- o": null} [2018-02-27 06:33:15.740] [DEBUG] agent.js - STATUS: 200 [2018-02-27 06:33:15.740] [DEBUG] agent.js - HEADERS: {"content-type": "application/json", "con- tent-length": "430", "date": "Tue, 27 Feb 2018 11:33:15 GMT", "connection": "close"} [2018-02-27 06:33:15.740] [DEBUG] agent.js - BODY: {"returnCode": "Success", "result": "[{\\"TxId\\": \"562001401636884cebb3db925d167d25a67476793d7a3db07a784fb7276318c0\", \\"Value\\": {\\"docType\\": \"vehiclePart\", \\"serialNumber\\": \"abg1234\", \\"assembler\\": \"panama-parts\", \\"assemblyDate\\": 1502688979, \\"name\\": \"airbag 2020\", \\"owner\\": \"detroit auto\", \\"recall\\": false, \\"recallDa- te\\": 1502688979}, \\"Timestamp\\": \"2018-02-12 17:57:22.906 +0000 UTC\", \\"IsDelete\\": false}]", "info": null} </pre>
5.05	View transaction details	Selecting an item in the grid displays its history. Selecting a transaction displays the items historical state in detail.

PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description										
		<p>Vehicle Parts</p> <table border="1"> <thead> <tr> <th>serialNumber</th> <th>assembler</th> <th>assemblyDate</th> <th>name</th> <th>owner</th> </tr> </thead> <tbody> <tr> <td>abg1234</td> <td>panama-parts</td> <td>2017-08-14T05:36:19.000Z</td> <td>airbag 2020</td> <td>detroit auto</td> </tr> </tbody> </table>  <pre> { "VehicleParts": ["abg1234": [{ "TxId": "760484157b097134191", "Value": { "docType": "vehiclePart", "serialNumber": "abg1234", "assembler": "panama-parts", "assemblyDate": 1502688979, "name": "airbag 2020", "owner": "detroit auto", "recall": false, "recallDate": 1502688979 }, "Timestamp": "2018-06-18 13:13:13", "IsDelete": "false" }, { "TxId": "0be1a03ef83ed590bedf", "Value": { "docType": "vehiclePart", "serialNumber": "abg1234", "assembler": "panama-parts", "assemblyDate": 1502688979, "name": "airbag 2020", "owner": "detroit auto", "recall": true, "recallDate": 1502688979 }, "Timestamp": "2018-10-02 07:41:49.211 +0000 UTC", "IsDelete": "false" }]] } </pre>	serialNumber	assembler	assemblyDate	name	owner	abg1234	panama-parts	2017-08-14T05:36:19.000Z	airbag 2020	detroit auto
serialNumber	assembler	assemblyDate	name	owner								
abg1234	panama-parts	2017-08-14T05:36:19.000Z	airbag 2020	detroit auto								

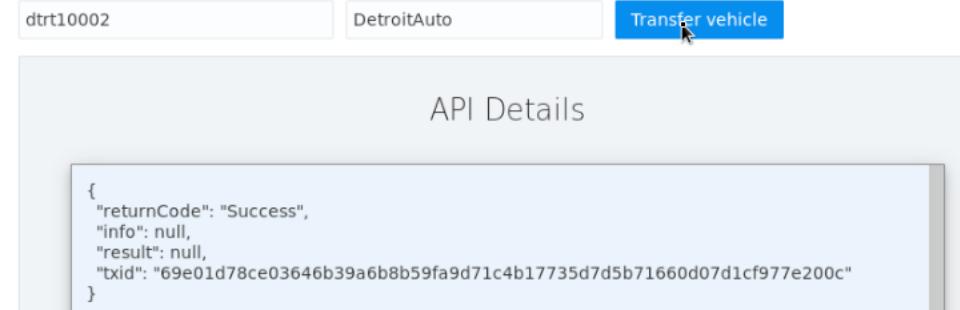
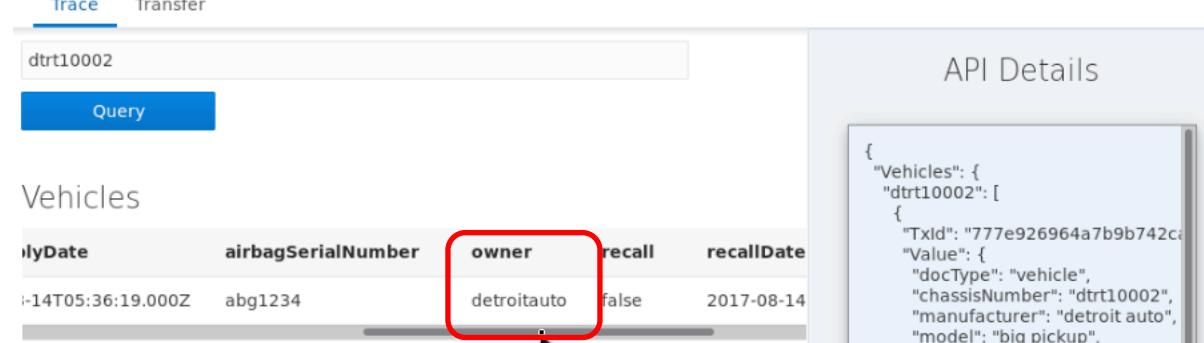
PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
5.06		Executing a transaction between the manufacturer and one dealer via the sample app. The sample apps can also be used to demonstrate how to change ownership of a vehicle. We are going to transfer a vehicle "dtrt10001" from "Sam's Dealership" to "Detroit Auto".
5.07	<p>6. On your Blockchain VM open the 'Home' directory and select 'Documents/environments'</p> <p>7. Open the 'SamDealer' directory and right-mouse click in the background</p> <p>8. Select 'Open Terminal' from the menu.</p> <p>9. In the Terminal window enter 'npm start'.</p> <p>10. The sample app will start.</p> <p>Note: You can monitor the process in the terminal window.</p>	 

PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description
5.08	<p>11. Open the web browser and 12. type in the following URL: http://localhost:7020. This will open the sample app UI for the Car dealer.</p>	
5.09	<p>Let's check if the vehicle dtrt10002 belongs actually to Sam</p> <ol style="list-style-type: none"> Enter 'dtrt10002' in the Query field and Click Query The output states that the vehicle actually belong to Sam. 	
5.10	<p>Now, we want to transfer the vehicle back to the manufacturer</p> <ol style="list-style-type: none"> Select the 'Transfer' tab Login using username 'SamDealer' and any password Click 'Login' Enter 'dtrt10002' as chassis number Enter 'DetroitAuto' as new vehicle owner 	

PaaS – BlockChain Cloud Service (BCS)

S.No.	Action	Description										
	<p>6. Click 'Transfer Vehicle'</p> <p>7. Once the transaction has been verified and processed, the result is displayed in the app.</p> <p>8. A quick check with the Query feature confirms the successful transaction.</p>	 <p>The screenshot shows the 'Transfer vehicle' button being clicked. Below it, an 'API Details' box displays a JSON response indicating success:</p> <pre>{ "returnCode": "Success", "info": null, "result": null, "txid": "69e01d78ce03646b39a6b8b59fa9d71c4b17735d7d5b71660d07d1cf977e200c" }</pre>  <p>The screenshot shows the 'Query' tab selected. It displays a table of vehicle data, with the 'owner' column for the first row highlighted and circled in red. The table data is as follows:</p> <table border="1"> <thead> <tr> <th>vin</th> <th>airbagSerialNumber</th> <th>owner</th> <th>recall</th> <th>recallDate</th> </tr> </thead> <tbody> <tr> <td>1H14T05:36:19.000Z</td> <td>abg1234</td> <td>detroitauto</td> <td>false</td> <td>2017-08-14</td> </tr> </tbody> </table> <p>The 'owner' column value 'detroitauto' is circled in red.</p>	vin	airbagSerialNumber	owner	recall	recallDate	1H14T05:36:19.000Z	abg1234	detroitauto	false	2017-08-14
vin	airbagSerialNumber	owner	recall	recallDate								
1H14T05:36:19.000Z	abg1234	detroitauto	false	2017-08-14								

Congratulations**You completed Chapter 5 and have explored the Blockchain access through sample applications.**