
Oracle
BlockChain
Cloud Service
Cloud Test Drive Workshop
ORACLE
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. |

Contents:

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

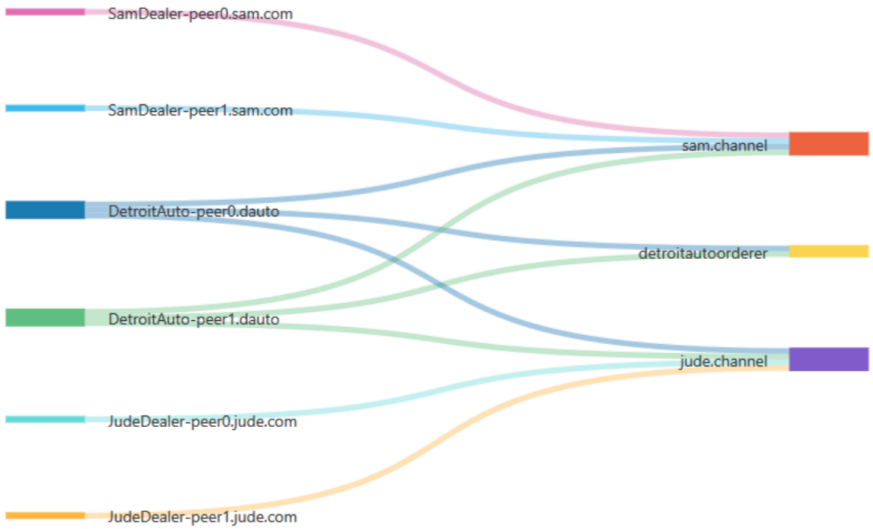
Oracle Blockchain Cloud Service is a new offering that is part of Oracle's comprehensive platform-as-a service (PaaS) portfolio. Delivered by the world's most scalable, distributed transaction processing platform provider, Oracle Blockchain Cloud Service is the most comprehensive distributed ledger cloud 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.

Overview

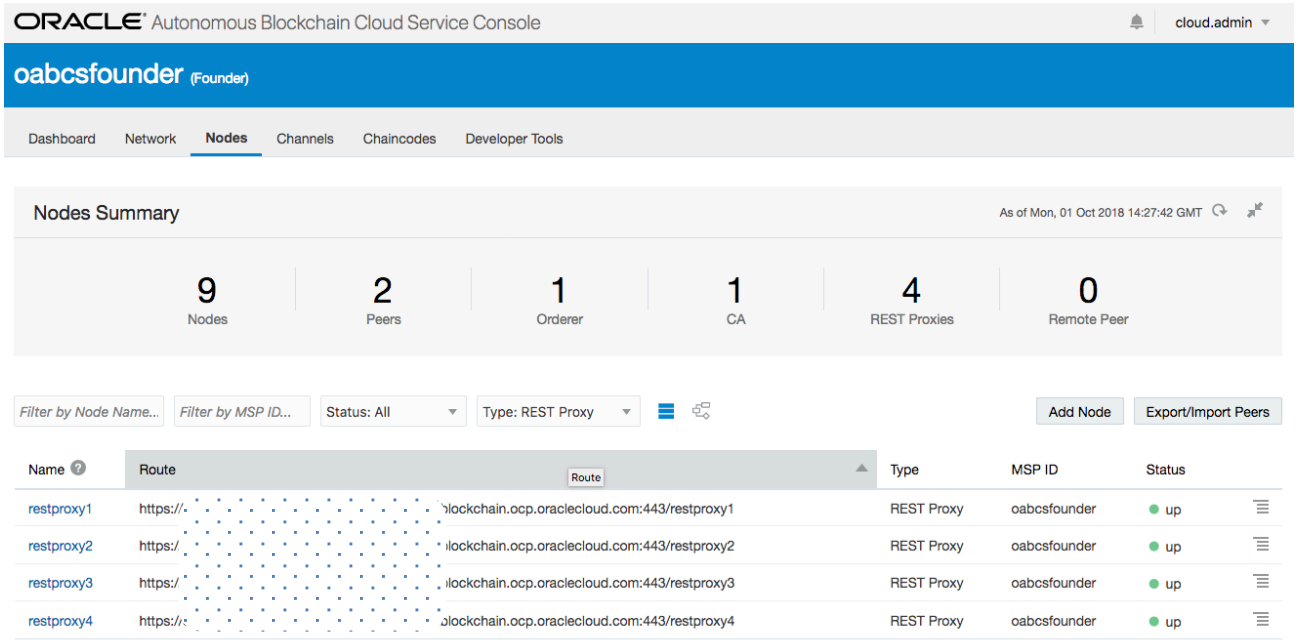
End-to-End Application Flow

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 OBCS REST API and via sample Business Applications.

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

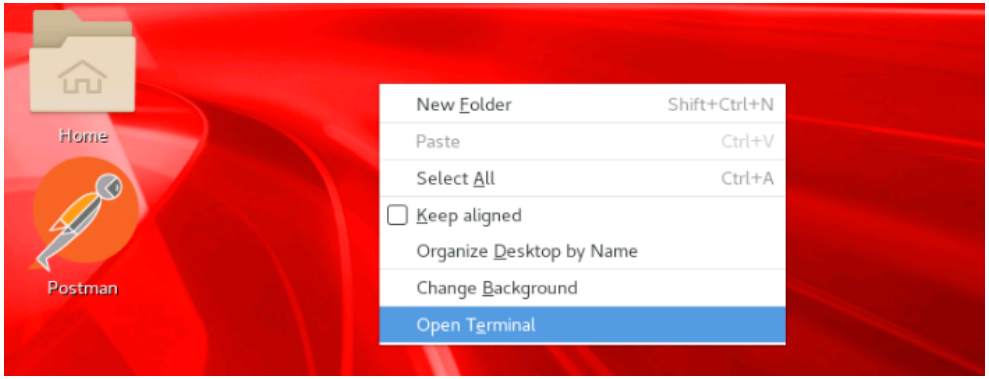
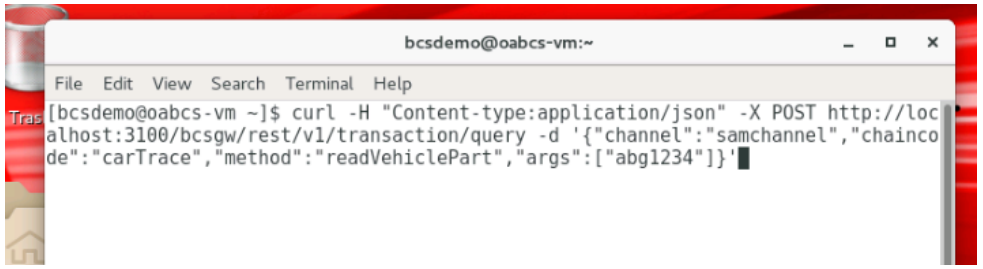
S.No.	Action	Description
Chapter 1: Recap and starting point		
1.00	Whether you come from the previous lab 'Creating a Blockchain network' or starting fresh with this lab, you'll find a working Blockchain network based on Hyperledger Fabric that will run in the Oracle Cloud on 3 instances (Blockchain Network participants) of Oracle Blockchain Cloud Service (OBCS). 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 provide with OBCS.  Now, we are going to work on this network and create transactions that will be recorded as blocks in to our Blockchain network. The OBCS 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 OBCS. We will begin with the REST API and some sample queries. 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.	
1.01	ChainCode / Smart Contract	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. 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

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".	
1.02	Looking into the ChainCode we can find the following Methods and Arguments which are defined for our **carTrace** sample: **NOTE: A full list of available Methods and Arguments can be seen in the carTrace.go file on the VM in the /Documents folder.**	**Method**	**Arguments**
		initVehiclePart (add a car part)	part id: string assembler: string assembly time: timestamp owner: string recall status: boolean recall time: timestamp
		initVehicle (add a car)	vehicle id: string manufacturer: string vehicle name: string assembly time: timestamp airbag serial: string owner: string recall status: Boolean recall time: timestamp
		transferVehiclePart (transfer ownership)	part id: string current owner: string new owner: string
		transferVehicle (transfer ownership)	vehicle id: string current owner: string new owner: string
		readVehiclePart (read records)	vehicle id: string part id: string
		getHistoryForRecord (read records)	part id: string
		deleteVehiclePart (delete records)	part id: string
		setPartRecallState (recall a part)	part id: string status: Boolean

S.No.	Action	Description
1.03	OBCS REST API	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. The REST API is available through a REST proxy on each Blockchain Network participant's instance.  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/devapplicationtasks.html

1.04	Blockchain transactions	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: Channel PartID Samchannel abg1234, abg1235, ser1236, win1237, bra1238 Judechannel abg1239, abg1240, whl1241, win1242, sen1243 Channel vehicleID Samchannel dtrt10001, dtrt10002
Congratulation **You completed Chapter 1**		

Chapter 2: Working with the Blockchain – REST API basics

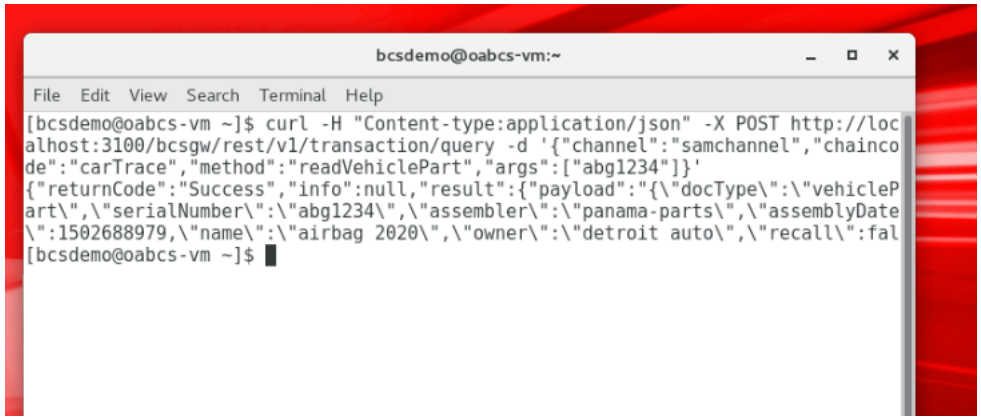
2.00	In this chapter we will begin with the REST API and try some sample queries against our Blockchain Network. You can work against the ledger (**Query, Invocation**) by making REST calls using the terminal or apps like Postman. What you can do against the ledger is defined in the instantiated chaincode (see previous chapter).	
2.01	The default syntax for working on the Command Line looks like the following: **NOTE: The method and arguments depend on your choice of method. See previous section.** **You could either do an Invocation or Query.**	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>*" }'
2.02	Let's do a query against the ledger for a specific car part. 1. Open a terminal window 2. Enter the following sequence at the cursor position.	 

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).

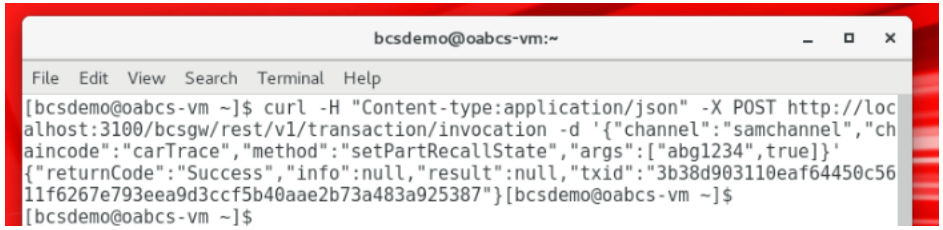
```
{"returnCode":"Success","info":null,"result":{
"payload":
"{\"docType\":\"vehiclePart\",
\"serialNumber\":\"abg1234\",
\"assembler\":\"panama-parts\",
\"assemblyDate\":1502688979,
\"name\":\"airbag 2020\",
\"owner\":\"detroit auto\",
\"recall\":false,
\"recallDate\":1502688979}",
"encode":"UTF-8"}}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



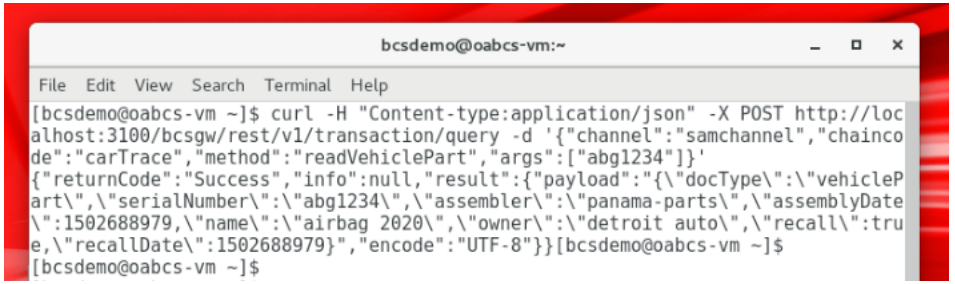
Congratulation

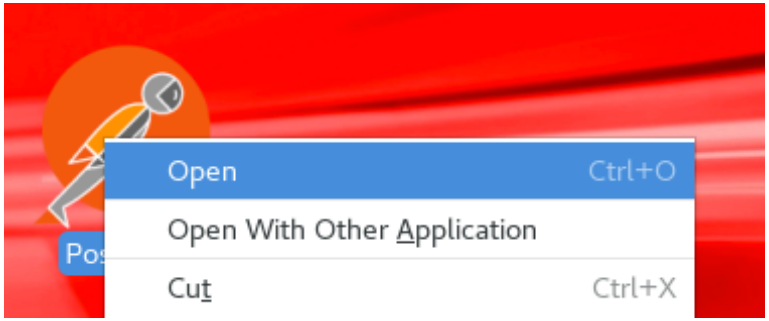
You completed Chapter 2 and have accessed the Blockchain Ledger from a Terminal window.

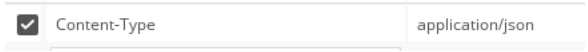
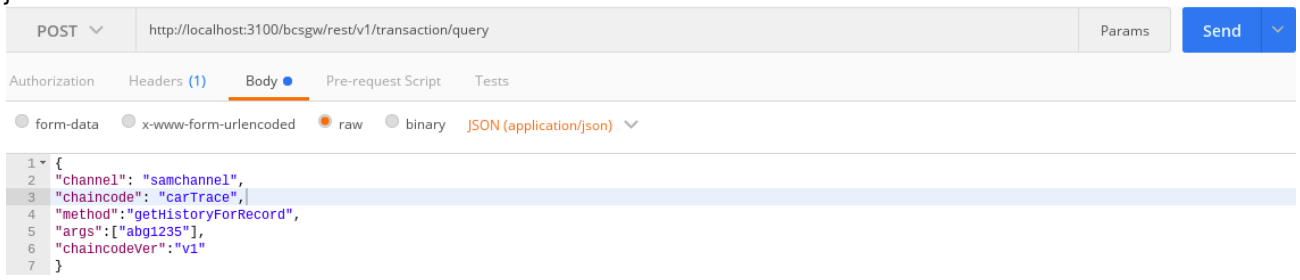
Chapter 3: Working with the Blockchain – REST API making changes

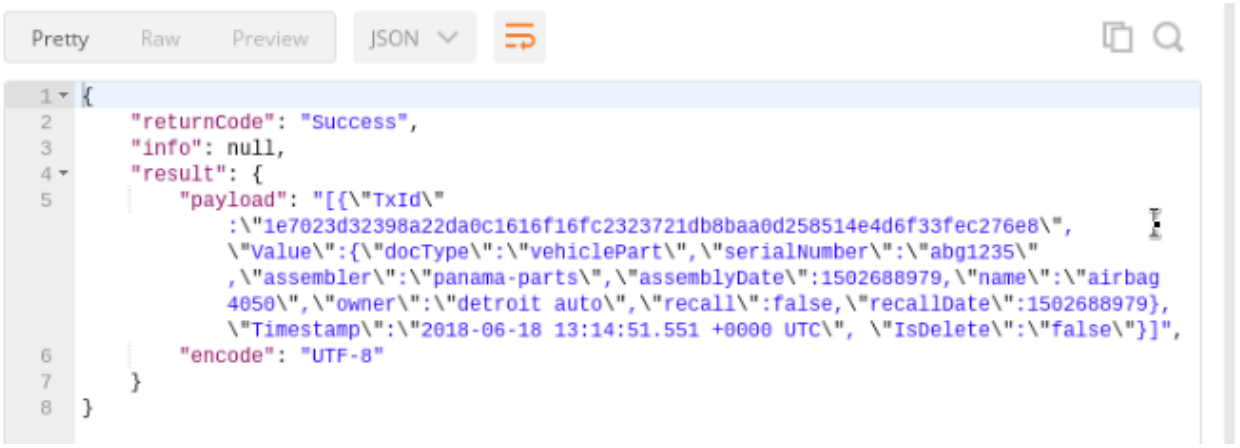
3.00	Now we want to make use of a feature in our Chaincode / SmartContract. Assuming that the Car manufacturer would like to recall a specific part delivered to the dealer. This part (record) must be changed and marked as recalled. As we cannot change the Blockchain, we have to add a new block where we append the existing information with the recall.	
3.01	We are going to "**invoke**" a method on the "carTrace" chaincode that will effectively mark a particular part as recalled via a REST call. 1. Type the following sequence into your terminal window. 2. Hit 'Return' to submit the command.	`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]}'`
3.02	3. The expected outcome should look like this **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.**	`{"returnCode":"Success","info":null,"result":null,"txid":"583f3ca55b9f2860875b8978e386611b476892c348440a248f07e98dbdad419d"}` 
3.03	In order to verify the change (recall), let's execute the query from **Chapter 2** again:	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"]}'

3.04	This is the result of this query:	`{"returnCode":"Success","info":null,"result":`
		`{"payload":"`
		`{\"docType\":\"vehiclePart\",`
		`\"serialNumber\":\"abg1234\",`
		`\"assembler\":\"panama-parts\",`
		`\"assemblyDate\":1502688979,`
		`\"name\":\"airbag 2020\",`
		`\"owner\":\"detroit auto\",`
		`\"recall\":true,`

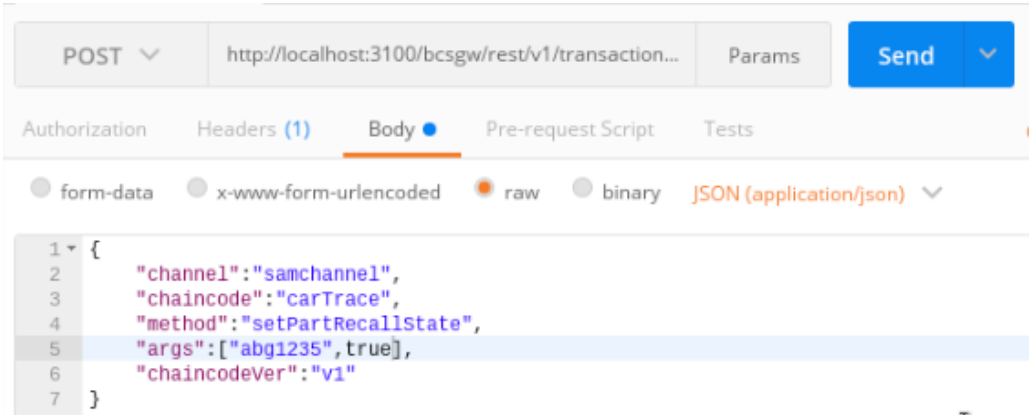
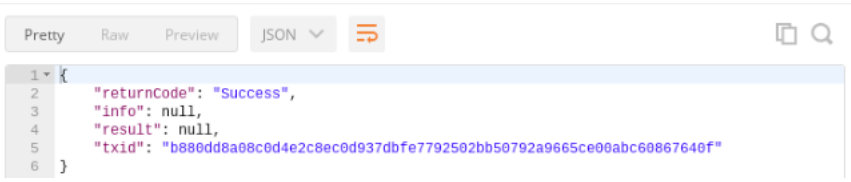
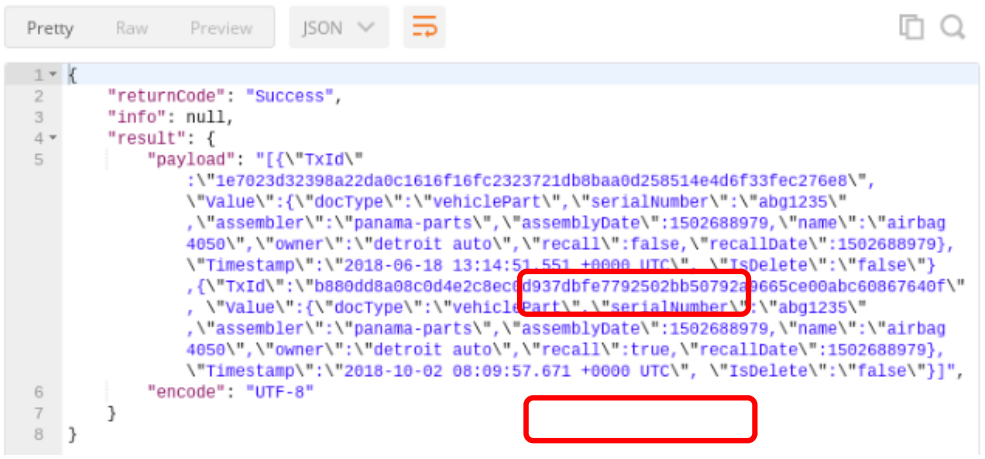
<table>
<tr><td></td><td></td><td>\"recallDate\":1502688979}",
"encode":"UTF-8"}}
 
</td></tr>
<tr><td>3.05</td><td colspan="2">Each transaction in the Blockchain will be appended to the end of the previous transaction forming a chain of blocks. If we would like to see changes associated to an item we have to create a query against the transaction history.</td></tr>
<tr><td>3.06</td><td>1. In the terminal window enter the following sequence</td><td>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"]}'</td></tr>
<tr><td>3.07</td><td>2. This time the result is more complex as it contains all transaction details for the specific partID.</td><td>File Edit View Search Terminal Help
[bcsdemo@oabcs-vm ~]$ 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"]}'
{"returnCode":"Success","info":null,"result":{"payload":"[{\"TxId\":\"760484157b097134191ab7441380a24bf19eacf66c266cee79646fdc905e14ac\", \"Value\":{\"docType\":\"vehiclePart\",\"serialNumber\":\"abg1234\",\"assembler\":\"panama-parts\",\"assemblyDate\":1502688979,\"name\":\"airbag 2020\",\"owner\":\"detroit auto\",\"recall\":false,\"recallDate\":1502688979}, \"Timestamp\":\"2018-06-18 13:13:43.344 +0000 UTC\", \"IsDelete\":\"false\"},{\"TxId\":\"0be1a03ef83ed590bedf1944f8b549d9ba71bb8a6d551d817a3833cd5eb7723c\", \"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\"},{\"TxId\":\"3b38d903110eaf64450c5611f6267e793eea9d3ccf5b40aae2b73a483a925387\", \"Value\":{\"docType\":\"vehiclePart\",\"serialNumber\":\"abg1234\",\"assembler\":\"panama-parts\",\"assemblyDate\":1502688979,\"name\":\"airbag 2020\",\"owner\":\"detroit auto\",\"recall\":true,\"recallDate\":1502688979}, \"Timestamp\":\"2018-10-02 07:43:44.189 +0000 UTC\", \"IsDelete\":\"false\"},{\"TxId\":\"f82033db48de1be0aa65f3d1bc423805a3fffc12ec86106aac0646bf179a9e2a\", \"Value\":{\"docType\":\"vehiclePart\",\"serialNumber\":\"abg1234\",\"assembler\":\"panama-parts\",\"assemblyDate\":1502688979,\"name\":\"airbag 2020\",\"owner\":\"detroit auto\",\"recall\":true,\"recallDate\":1502688979}, \"Timestamp\":\"2018-10-02 07:44:17.105 +0000 UTC\", \"IsDelete\":\"false\"},{\"TxId\":\"9b00672b2f937d8ed2b96f826b75c001939f3a4a5664b3cae11b8aac30545a23\", \"Value\":{\"docType\":\"vehiclePart\",\"serialNumber\":\"abg1234\",\"assembler\":\"panama-parts\",\"assemblyDate\":1502688979,\"name\":\"airbag 2020\",\"owner\":\"detroit auto\",\"recall\":false,\"recallDate\":1502688979}, \"Timestamp\":\"2018-10-02 07:45:42.993 +0000 UTC\", \"IsDelete\":\"false\"}]","encode":"UTF-8"}}[bcsdemo@oabcs-vm ~]$</td></tr>
<tr><td colspan="3"><u>Congratulations</u>
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.</td></tr>
</table>

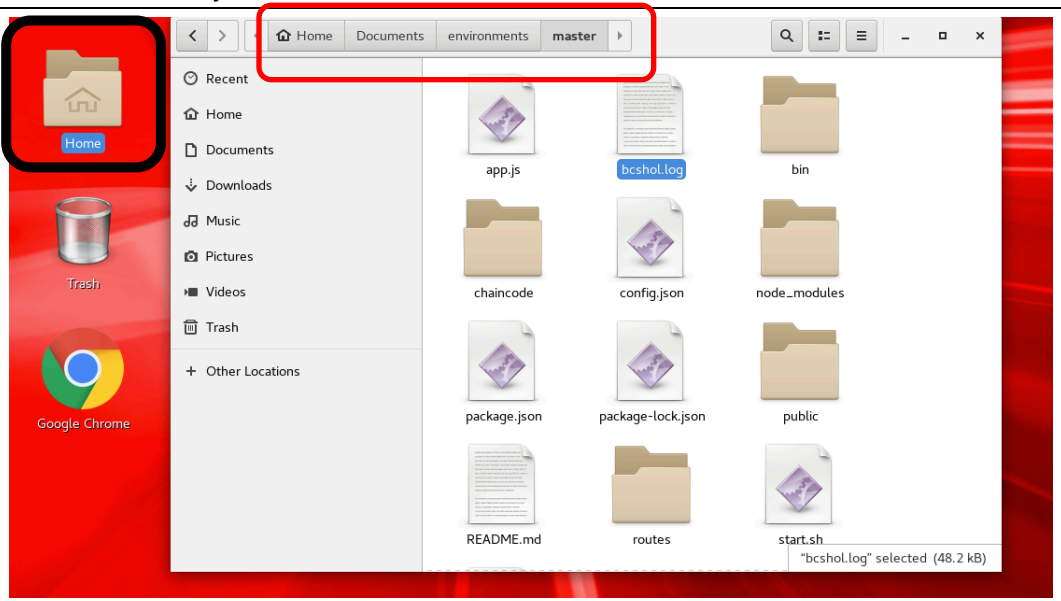
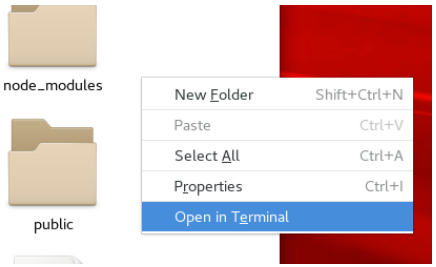
S.No.	Action	Description
Chapter 4: Working with the Blockchain – Using Postman		
4.00	Another way to interact with the Blockchain Cloud Service is to use a tool that is using the REST API to connect to the service and provides a User Interface. Such a tool is e.g. Postman (https://www.getpostman.com/). We are using Postman to query the Blockchain using the REST API. 1. Launch Postman from the VMs Desktop:	

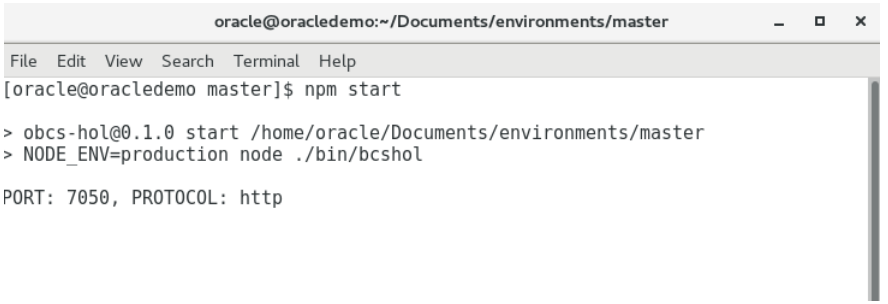
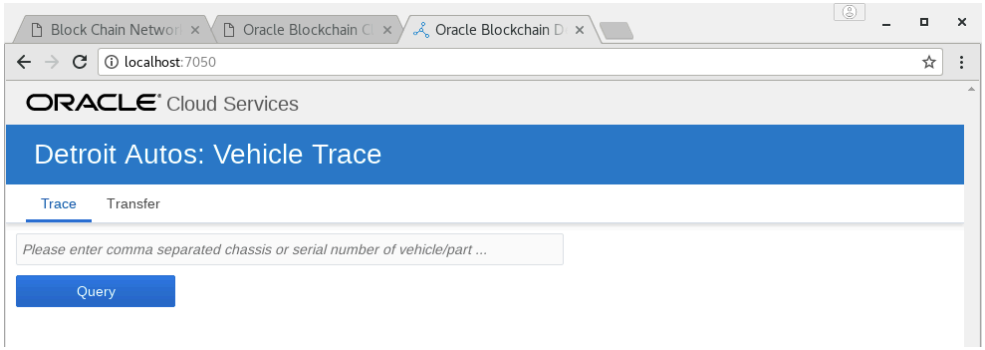
4.01	Query some details from the Blockchain Network.
	We will check if a particular record has been recorded at the Blockchain.
	2. In Postman, enter the following details:
	
	Note: We are using another method 'getHistory For Record' in this example.
	
	3. Click 'Send'
		Modus: POST
		URL: http://localhost:3100/bcsgw/rest/v1/transaction/query
		
		Headers: 
		
		Body – select 'RAW' and paste:
		{
		"channel":"samchannel",
		"chaincode":"carTrace",
		"method":"getHistoryForRecord",
		"args":["abg1235"],
		"chaincodeVer":"v1"
		}
		 

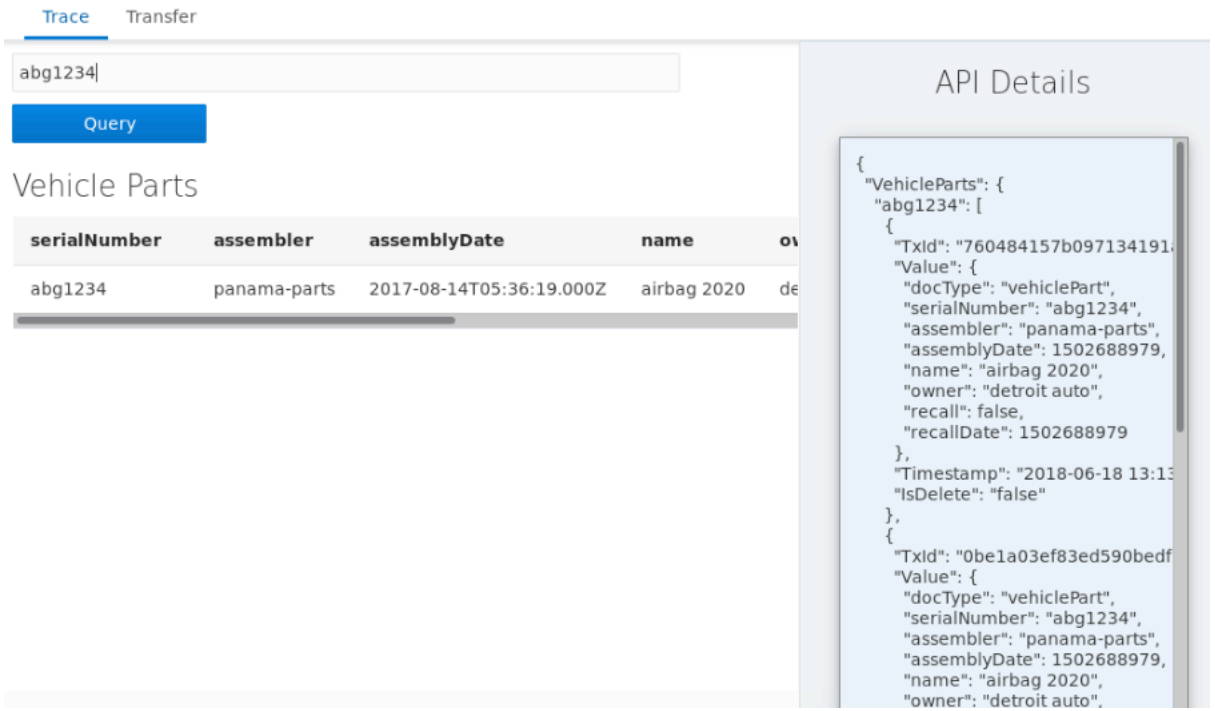
S.No.	Action	Description
4.02	In the query result, we can see that the record (abg1235) exists in the Blockchain with the transactionID and the transaction details. All this is info that has been submitted to the record when the ChainCode / Smart Contract has been executed.	

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:
		Modus: POST
		URL: http://*yourRESTProxy_endpoint:PORT*/bcsgw/rest/v1/transaction/invocation
		
		Headers: 
		
		Body – select 'RAW' and paste:
		{
		"channel":"samchannel",
		"chaincode":"carTrace",
		"method":"setPartRecallState",
		"args":["abg1235",true],
		"chaincodeVer":"v1"
		}

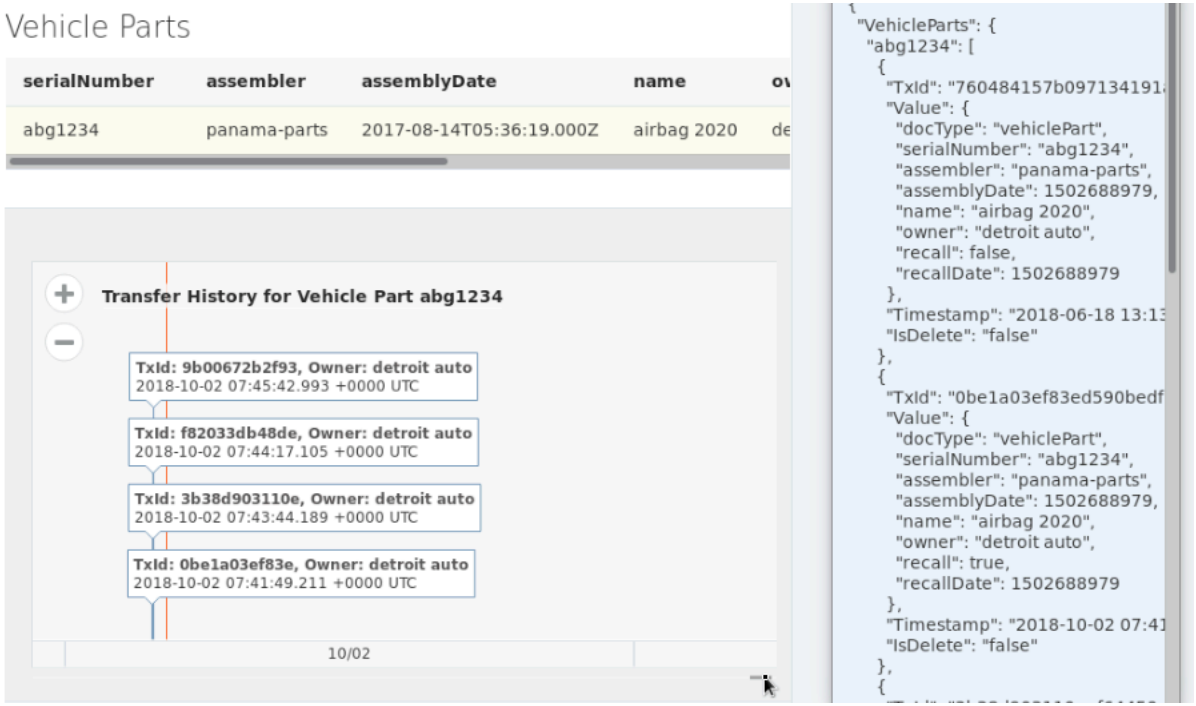
S.No.	Action	Description
		
4.04	And the expected outcome in Postman.	
4.05	To verify that the information has been updated we will run the 'getHistoryFor Record' again. See the result in Postman.	
Congratulations **You have just learned a different way to interact with the Blockchain Cloud Servcice using Postman and the REST API**		

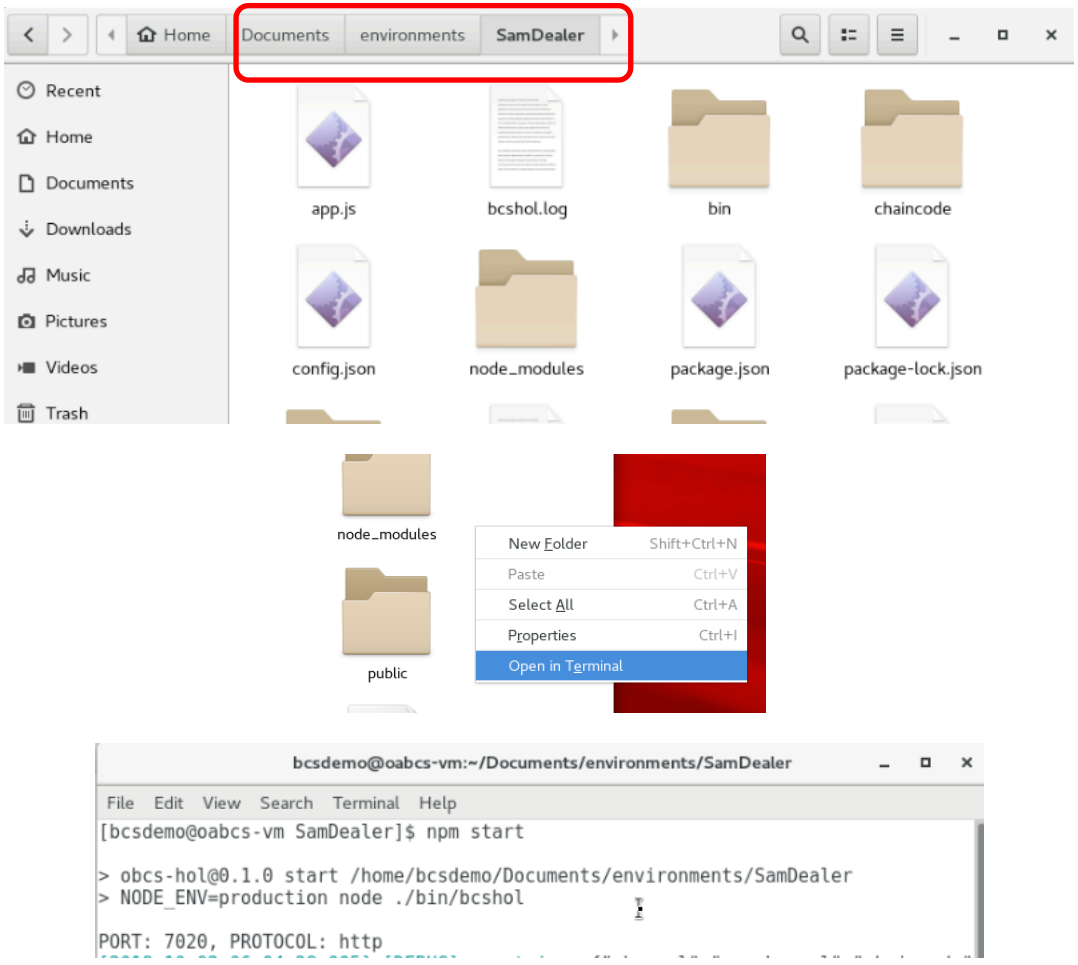
S.No.	Action	Description
Chapter 5: Working with the Blockchain – Sample Apps		
5.00	In a real business scenario, Blockchain will be embedded in Business Applications that trigger transactions on the Blockchain. For demo purpose we have built a sample app that mimics such behaviour and let you work with the Oracle Blockchain Cloud Service.	
5.01	1. On your Blockchain VM open the 'Home' directory and select 'Documents/environments' 2. Open the 'master' directory and right-mouse click in the background 3. Select 'Open Terminal' from the menu.	 

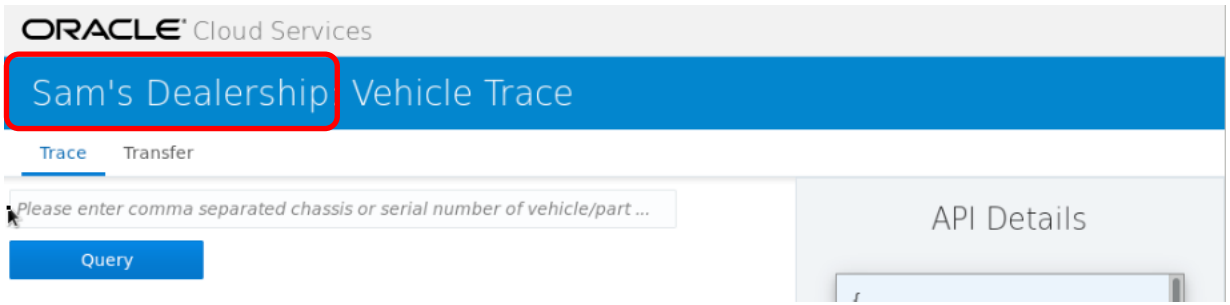
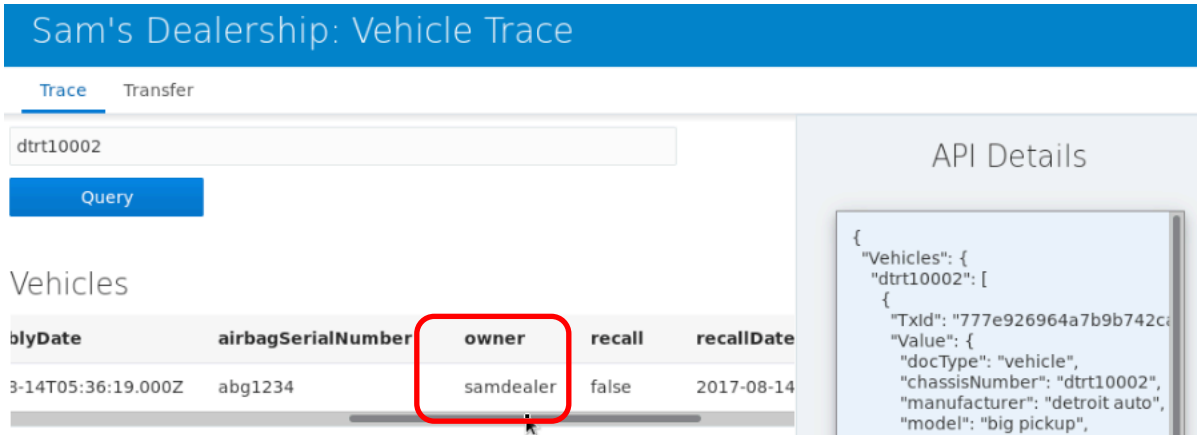
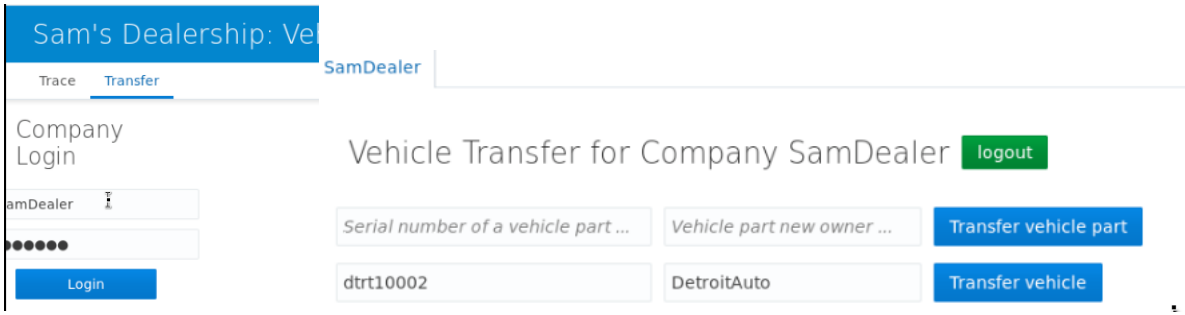
S.No.	Action	Description
	4. In the Terminal window enter 'npm start'. 5. The sample app will start. ***Note: You can monitor the process in the terminal window.***	
5.02	1. Open the webbrowser and 2. type in the following URL: http://localhost:7050 3. This will open the sample app UI for the Car manufacturer.	
5.03	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.	

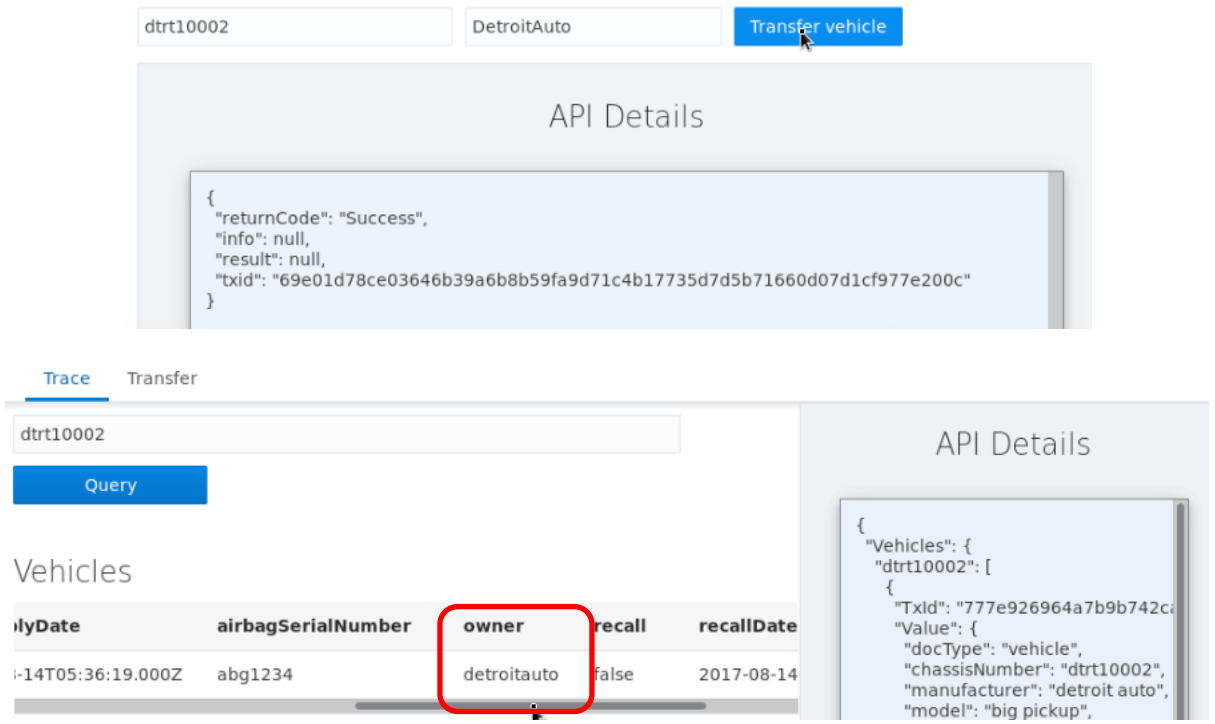
S.No.	Action	Description
5.04	Working with the Blockchain. Querying the ledger for automotive parts. 1. Enter some valid part numbers in the Query field **Valid part numbers:** abg1234 abg1235 2. You can see the part details alongside with the REST API result set. 3. In our sample app the query feature always retrieves the full history of the partID using the 'getHistoryForRecord' method. Again, the terminal window provides the raw output.	

S.No.	Action	Description
		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}
5.05	View transaction details	Selecting an item in the grid displays its history. Selecting a transaction displays the items historical state in detail.

S.No.	Action	Description
		

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	6. On your Blockchain VM open the 'Home' directory and select 'Documents/environments' 7. Open the 'SamDealer directory and right-mouse click in the background 8. Select 'Open Terminal' from the menu. 9. In the Terminal window enter 'npm start'. 10. The sample app will start. **Note: You can monitor the process in the terminal window.**	

S.No.	Action	Description
5.08	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.	
5.09	Let's check if the vehicle dtrt10002 belongs actually to Sam 1. Enter 'dtrt10002' in the Query field and 2. Click Query 3. The output states that the vehicle actually belong to Sam.	
5.10	Now, we want to transfer the vehicle back to the manufacturer 1. Select the 'Transfer' tab 2. Login using username 'SamDealer' and any password 3. Click 'Login' 4. Enter 'dtrt10002' as chassis number 5. Enter 'DetroitAuto' as new vehicle owner	

S.No.	Action	Description
	6. Click 'Transfer Vehicle' 7. Once the transaction has been verified and processed, the result is displayed in the app. 8. A quick check with the Query feature confirms the successful transaction.	
	Congratulations **You completed Chapter 5 and have explored the Blockchain access through sample applications.**	