

Oracle Content & Experience Cloud Service

Workshop

Contents:

[Demo Attributes 2](#_Toc509396936)

[Oracle Blockchain Cloud Service 2](#_Toc509396937)

[Overview 3](#_Toc509396938)

[Chapter 1: Working with the Blockchain – REST API basics 4](#_Toc509396939)

[Chapter 2: Working with the Blockchain – making changes 6](#_Toc509396940)

[Chapter 3: Working with the Blockchain – Sample Apps 8](#_Toc509396941)

## Demo Attributes

|  |  |
| --- | --- |
| **Product(s)** | Oracle BlockChain Cloud Service |
| **Date last updated** | March 2018 |
| **Author(s)** | Jens Lusebrink |
| **Demo Title(s)** | PaaS – BlockChain Cloud Service (BCS) |

## 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: Working with the Blockchain – REST API basics | | |
| 1.00 | In the previous lab we have learned how to create a Blockchain network of 3 parties (1 founder, 2 participants) around a fictive Car Manufacturer/Dealer scenario.  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. | We are now using Postman (<https://www.getpostman.com/>) to query the Blockchain using the REST API.  Launch Postman from the VMs Desktop: |
| 1.01 | Query some details from the Blockchain Network.  We will check if a particular record has been recorded at the Blockchain, | In Postman, enter the following details:  **Modus:** POST  **URL:** http://localhost:<yourPortNumber>/bcsgw/rest/v1/transaction/query  **Headers:**  **Body – select ‘RAW’ and paste:**  {  "channel":"sam.channel",  "chaincode":"carTrace",  "method":"getHistoryForRecord",  "args":["abg1234"],"chaincodeVer":"v1"  }    Click ‘Send’. |
| 1.02 | 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.  **NOTE: In our case, when we had executed the *initledger* script.** |  |

|  |  |  |
| --- | --- | --- |
| Chapter 2: Working with the Blockchain – REST API making changes | | |
| 2.00 | 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. | Now we are going to “invoke” a method on the “carTrace” chaincode that will effectively mark a particular part as recalled via a REST call.  In Postman, change the URL to:  http://localhost:3100/bcsgw/rest/v1/transaction/invocation  In the Body, write:  {  “channel":"sam.channel",  "chaincode":"carTrace",  "method":"setPartRecallState",  "args":["abg1234",true],  "chaincodeVer":"v1"  } |
| 2.01 | Return code | Your return code should be a confirmation of this transaction:  {"returnCode":"Success","info":null,"transactionID":"04fc199cb7307622e1f74659e8dfaef4eed6554ae96aa33fa34085375899357f"} |
| 2.02 | Verify | In order to verify the change (recall), let’s execute the query from above again:  <http://localhost:3100/bcsgw/rest/v1/transaction/query>  {"channel":"sam.channel","chaincode":"carTrace","method":"getHistoryForRecord","args":["abg1234"],"chaincodeVer":"v1"}  **RESULT:**  {  "returnCode": "Success",  "result": "[{\"TxId\":\"a05505990f67e45d33666091552d8dfc3d7752f3a99ff67322cdea42722f2f90\", \"Value\":{\"docType\":\"vehiclePart\",  \"serialNumber\":\"abg1234\",  \"assembler\":\"panama-parts\",  \"assemblyDate\":1502688979,  \"name\":\"airbag 2020\",  \"owner\":\"detroitauto\",  \"recall\":false,  \"recallDate\":1502688979},  \"Timestamp\":\"2018-03-21 09:58:09.97 +0000 UTC\",  \"IsDelete\":\"false\"},  {\"TxId\":\"70cfec93b4c6b9719caf3925ff77f966a22f7d606cdc52e2522f0601878ba508\", \"Value\":{\"docType\":\"vehiclePart\",  \"serialNumber\":\"abg1234\",  \"assembler\":\"panama-parts\",  \"assemblyDate\":1502688979,  \"name\":\"airbag 2020\",  \"owner\":\"detroitauto\",  \"recall\":true,  \"recallDate\":1502688979},  \"Timestamp\":\"2018-03-21 10:53:55.787 +0000 UTC\",  \"IsDelete\":\"false\"}]",  "info": null  } |
| **Congratulation**  **You completed Chapter 2** | | |

|  |  |  |
| --- | --- | --- |
| Chapter 3: Working with the Blockchain – Sample Apps | | |
| 3.00 | Summary | In real business scenarios, Blockchain will be embedded in Business Applications that trigger transactions on the Blockchain. For demo purpose we have build a sample app that mimics such behaviour and let you work with the Oracle Blockchain Cloud Service. |
| 3.02 | Sample App – starting | On your OBCS VM, double click the ‘Home’ icon and drill down to the “Home/Documents/environments/master directory.    Right-mouse click in the directory and select ‘Open in Terminal’    In the Terminal window enter ‘npm start’.  The sample app will start.  ***Note: You can monitor the process in the terminal window.*** |
| 3.03 | Accessing the sample app. | Open the Chrome webbrowser and tyoe in the following URL: <http://localhost:7050>  This will open the sample app UI. |
| 3.04 | Working with the Blockchain. Querying the ledger for automotive parts.  **Valid part numbers:**  abg1234  abg1235  dtrt10001  dtrt10002  whl1241  win1242 | The sample application includes simple web interfaces interacting directly with BCS 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 BCS using REST.  The dealers push transactions to a private channel shared with the manufacturer e.g. “sam.channel”, 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. Open the  Query the ledger by entering some part numbers in the field on the ‘Trace’ page.    You can see the part details alongside with the REST API result set.  Again, the terminal window provide the raw output. |
|  | View transaction details | Selecting an item in the grid displays its history. Selecting a transaction displays the items historical state in detail. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | | |
| **Congratulation**  **You completed Chapter 3** | | |