Contents

[1. Deploy 2](#_Toc457382389)

[2. Invoke 3](#_Toc457382390)

[2.1 Create Vehicle 3](#_Toc457382391)

[2.2 Authority to Manufacturer 4](#_Toc457382392)

[2.3 Manufacturer to Private 5](#_Toc457382393)

[2.4 Private to Private 6](#_Toc457382394)

[2.5 Private to Lease Company 7](#_Toc457382395)

[2.6 Lease Company to Private 8](#_Toc457382396)

[2.7 Private to Scrap Merchant 9](#_Toc457382397)

[2.8 Update Colour 10](#_Toc457382398)

[2.9 Update Make 11](#_Toc457382399)

[2.10 Update Model 12](#_Toc457382400)

[2.11 Update Registration 13](#_Toc457382401)

[2.12 Update VIN 14](#_Toc457382402)

[2.13 Scrap Vehicle 15](#_Toc457382403)

[3. Query 17](#_Toc457382404)

[3.1 Get Vehicle Details 17](#_Toc457382405)

[3.2 Get Vehicles 19](#_Toc457382406)

[4. Glossary 22](#_Toc457382407)

# Deploy

##### Server Side API Call

POST /blockchain/chaincode/vehicles

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "deploy",

"params": {

"type": 1,

"chaincodeID": {

"path": <chaincode\_path>

},

"ctorMsg": {

"function": "init",

"args": [

<api\_url>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

None.

##### Description

Deploys the chaincode located at the path that was supplied (e.g. https://github.com/IBM-Blockchain/car-lease-demo/Chaincode/vehicle\_code), calling the init function which stores in the world state a blank array of V5C IDs and the peer address supplied which is used to get eCerts.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<chaincode\_name>"

},

"id": <id>

}

# Invoke

## Create Vehicle

##### Server Side API Call

POST /blockchain/assets/vehicles

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "invoke",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "create\_vehicle",

"args": [

<v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

* The caller must be a Regulator.
* <v5c\_ID> passed must consist of 2 letters followed by 7 numbers and be unique.

##### Description

The function takes a <v5c\_ID> and creates a new blank vehicle JSON object to store in the world state with the ID of <v5c\_ID>.

JSON Object:

{"v5cID": "<v5c\_ID>", "VIN": 0, "make": "UNDEFINED",

"model": "UNDEFINED", "reg": "UNDEFINED", "owner": <caller>, "colour": "UNDEFINED", "leaseContractID": "UNDEFINED",

"status": 0, "scrapped": false}

The function checks whether the <v5c\_ID> has already been used and if not writes the JSON to the world state with the <v5c\_ID> as the key. It also appends the <v5c\_ID> to the array of V5C IDs and updates that in the world state.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<transaction\_id>"

},

"id": <id>

}

## Authority to Manufacturer

##### Server Side API Call

PUT /blockchain/assets/vehicles/<v5c\_ID>/owner

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "invoke",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "authority\_to\_manufacturer",

"args": [

<recipient>, <v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

* Caller must be a Regulator.
* Recipient must be a Manufacturer.
* Vehicle must be owned by the caller.
* Vehicle must not be scrapped.
* Vehicle must have a state of 0.
* A vehicle with the <v5c\_ID> must exist in the world state.

##### Description

If the conditions are met then the vehicle is transferred from the Authority to a Manufacturer. This is done by updating the JSON stored with the key <v5c\_ID> in the world state so that the owner field is the Recipient passed as an argument. The vehicle's status is also updated in the JSON to be 1 to show it is in the state of manufacture.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<transaction\_id>"

},

"id": <id>

}

## Manufacturer to Private

##### Server Side API Call

PUT /blockchain/assets/vehicles/<v5c\_ID>/owner

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "invoke",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "manufacturer\_to\_private",

"args": [

<recipient>, <v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

* Caller must be a Manufacturer.
* Recipient must be a Private Entity.
* Vehicle must be owned by the caller.
* Vehicle must not be scrapped.
* Vehicle must have a state of 1.
* A vehicle with the <v5c\_ID> must exist in the world state.

##### Description

If the conditions are met, then the vehicle is transferred from the Manufacturer to a Private Entity. This is done by updating the JSON stored with the key <v5c\_ID> in the world state so that the owner field is the Recipient passed as an argument. The vehicle's status is also updated in the JSON to be 2 to show it is in the state of private ownership.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<transaction\_id>"

},

"id": <id>

}

## Private to Private

##### Server Side API Call

PUT /blockchain/assets/vehicles/<v5c\_ID>/owner

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "invoke",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "private\_to\_private",

"args": [

<recipient>, <v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

* Caller must be a Private Entity.
* Recipient must be a Private Entity.
* Vehicle must be owned by the caller.
* Vehicle must not be scrapped.
* Vehicle must have a state of 2.
* A vehicle with the <v5c\_ID> must exist in the world state.

##### Description

If the conditions are met then the vehicle is transferred from the Private Entity to another Private Entity. This is done by updating the JSON stored with the key <v5c\_ID> in the world state so that the owner field is the Recipient passed as an argument.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<transaction\_id>"

},

"id": <id>

}

## Private to Lease Company

##### Server Side API Call

PUT /blockchain/assets/vehicles/<v5c\_ID>/owner

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "invoke",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "private\_to\_lease\_company",

"args": [

<recipient>, <v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

* Caller must be a Private Entity.
* Recipient must be a Lease Company.
* Vehicle must be owned by the caller.
* Vehicle must not be scrapped.
* Vehicle must have a state of 2.
* A vehicle with the <v5c\_ID> must exist in the world state.

##### Description

If the conditions are met then the vehicle is transferred from the Private Entity to a Lease Company. This is done by updating the JSON stored with the key <v5c\_ID> in the world state so that the owner field is the Recipient passed as an argument.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<transaction\_id>"

},

"id": <id>

}

## Lease Company to Private

##### Server Side API Call

PUT /blockchain/assets/vehicles/<v5c\_ID>/owner

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "invoke",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "lease\_company\_to\_private",

"args": [

<recipient>, <v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

* Caller must be a Lease Company.
* Recipient must be a Private Entity.
* Vehicle must be owned by the caller.
* Vehicle must not be scrapped.
* Vehicle must have a state of 2.
* A vehicle with the <v5c\_ID> must exist in the world state.

##### Description

If the conditions are met then the vehicle is transferred from the Lease Company to a Private Entity. This is done by updating the JSON stored with the key <v5c\_id> in the world state so that the owner field is the Recipient passed as an argument.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<transaction\_id>"

},

"id": <id>

}

## Private to Scrap Merchant

##### Server Side API Call

PUT /blockchain/assets/vehicles/<v5c\_ID>/owner

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "invoke",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "private\_to\_scrap\_merchant",

"args": [

<recipient>, <v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

* Caller must be a Private Entity.
* Recipient must be a Scrap Merchant.
* Vehicle must be owned by the caller.
* Vehicle must not be scrapped.
* Vehicle must have a state of 2.
* A vehicle with the <v5c\_ID> must exist in the world state.

##### Description

If the conditions are met then the vehicle is transferred from the Private Entity to a Scrap Merchant. This is done by updating the JSON stored with the key <v5c\_id> in the world state so that the owner field is the Recipient passed as an argument. The vehicle's status is also updated in the JSON to be 4 to show it is in the state of private ownership.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<transaction\_id>"

},

"id": <id>

}

## Update Colour

##### Server Side API Call

PUT /blockchain/assets/vehicles/<v5c\_ID>/colour

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "invoke",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "update\_colour",

"args": [

<string>, <v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

* Caller must NOT be a Scrap Merchant.
* Vehicle must be owned by caller.
* Vehicle must not be scrapped.
* A vehicle with the <v5c\_ID> must exist in the world state.

##### Description

If the conditions are met then the vehicle's colour is updated to match the value passed. This is done by updating the JSON stored with the key <v5c\_id> in the world state so that the colour field is the new value.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<transaction\_id>"

},

"id": <id>

}

## Update Make

##### Server Side API Call

PUT /blockchain/assets/vehicles/<v5c\_ID>/make

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "invoke",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "update\_make",

"args": [

<string>, <v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

* Caller must be a Manufacturer.
* Vehicle must be owned by the caller.
* Vehicle must not be scrapped.
* Vehicle must have a state of 1.
* A vehicle with the <v5c\_ID> must exist in the world state.

##### Description

If the conditions are met then the vehicle's make is updated to match the value passed. This is done by updating the JSON stored with the key <v5c\_id> in the world state so that the make field is the new value.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<transaction\_id>"

},

"id": <id>

}

## Update Model

##### Server Side API Call

PUT /blockchain/assets/vehicles/<v5c\_ID>/model

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "invoke",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "update\_model",

"args": [

<string>, <v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

* Caller must be a Manufacturer.
* Vehicle must be owned by the caller.
* Vehicle must not be scrapped.
* Vehicle must have a state of 1.
* A vehicle with the <v5c\_ID> must exist in the world state.

##### Description

If the conditions are met then the vehicle's model is updated to match the value passed. This is done by updating the JSON stored with the key <v5c\_id> in the world state so that the model field is the new value.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<transaction\_id>"

},

"id": <id>

}

## Update Registration

##### Server Side API Call

PUT /blockchain/assets/vehicles/<v5c\_ID>/reg

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "invoke",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "update\_registration",

"args": [

<string>, <v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

* Caller must NOT be a Scrap Merchant
* Vehicle must be owned by the caller.
* Vehicle must not be scrapped.
* A vehicle with the <v5c\_ID> must exist in the world state.

##### Description

If the conditions are met then the vehicle's registration is updated to match the value passed. This is done by updating the JSON stored with the key <v5c\_id> in the world state so that the registration field is the new value.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<transaction\_id>"

},

"id": <id>

}

## Update VIN

##### Server Side API Call

PUT /blockchain/assets/vehicles/<v5c\_ID>/VIN

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "invoke",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "update\_VIN",

"args": [

<VIN>, <v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

* Caller must be a Manufacturer
* Vehicle must be owned by the caller.
* Vehicle must not be scrapped.
* Vehicle must have a state of 1.
* Vehicle's VIN must be 0 i.e. not changed before.
* VIN must be a 15 digit number.
* A vehicle with the <v5c\_ID> must exist in the world state.

##### Description

If the conditions are met then the vehicle's VIN is updated to match the value passed. This is done by updating the JSON stored with the key <v5c\_id> in the world state so that the VIN field is the new value.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<transaction\_id>"

},

"id": <id>

}

## Scrap Vehicle

##### Server Side API Call

DELETE /blockchain/assets/vehicles/<v5c\_ID>

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "invoke",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "scrap\_vehicle",

"args": [

<v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

* Caller must be a Scrap Merchant
* Vehicle must be owned by the caller.
* Vehicle must not be scrapped.
* Vehicle must have a state of 4.
* A vehicle with the <v5c\_ID> must exist in the world state.

##### Description

If the conditions are met then the vehicle's VIN is updated to match the value passed. This is done by updating the JSON stored with the key <v5c\_id> in the world state so that the VIN field is the new value.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<transaction\_id>"

},

"id": <id>

}

# Query

## Get Vehicle Details

##### Server Side API Call

GET /blockchain/assets/vehicles/<v5c\_ID>

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "query",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "get\_vehicle\_details",

"args": [

<v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

* Caller must be owner OR a Regulator

##### Description

If the user is the owner or an authority the code returns the JSON Object with the key <v5c\_ID> from the world state.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "<vehicle\_json>"

},

"id": <id>

}

##### Errors

Output:

{

"jsonrpc": "2.0",

"error": {

"code": -32003,

"message": "Query failure",

"data": "Error when querying chaincode:   
 Error:Failed to execute transaction or   
 query(Permission Denied)"

},

"id": <id>

}

Reason:

The caller does not have permission to read the vehicle   
 requested.

Output:

{

"jsonrpc": "2.0",

"error": {

"code": -32003,

"message": "Query failure",

"data": "Error when querying chaincode:   
 Error:Failed to execute transaction or   
 query(QUERY: Error retrieving v5c   
 RETRIEVE\_V5C: Corrupt vehicle record)"

},

"id": <id>

}

Reason:

There is no data stored in the world state with the key   
 <v5c\_ID>.

Output:

{

"jsonrpc": "2.0",

"error": {

"code": -32003,

"message": "Query failure",

"data": " "Error when querying chaincode:   
 Error:Failed to execute transaction   
 or query(GET\_VEHICLE\_DETAILS: Invalid   
 vehicle object)

},

"id": <id>

}

Reason:

The data stored with the key <v5c\_ID> is not in the format   
 expected for a vehicle.

## Get Vehicles

##### Server Side API Call

GET /blockchain/assets/vehicles

##### Chaincode Spec

{

"jsonrpc": "2.0",

"method": "query",

"params": {

"type": 1,

"chaincodeID": {

"name": <chaincode\_name>

},

"ctorMsg": {

"function": "get\_vehicle\_details",

"args": [

<v5c\_ID>

]

},

"secureContext": "<caller>"

},

"id": <id>

}

##### Conditions

None.

##### Description

Goes through all vehicles that have been created and indexed and if the owner is the caller or the caller is an authority then it adds the vehicle JSON to an array to be returned.

##### Output

{

"jsonrpc": "2.0",

"result": {

"status": "OK",

"message": "[<vehicle\_json>, ... ,<vehicle\_json>]"

},

"id": <id>

}

##### Errors

Output:

{

"jsonrpc": "2.0",

"error": {

"code": -32003,

"message": "Query failure",

"data": "Error when querying chaincode:   
 Error:Failed to execute transaction or   
 query(Unable to get v5cIDs)"

},

"id": <id>

}

Reason:

The v5cIDs data is missing.

Output:

{

"jsonrpc": "2.0",

"error": {

"code": -32003,

"message": "Query failure",

"data": "Error when querying chaincode:   
 Error:Failed to execute transaction or   
 query(Corrupt V5C\_Holder)"

},

"id": <id>

}

Reason:

The v5cIDs data is not in the correct format.

Output:

{

"jsonrpc": "2.0",

"error": {

"code": -32003,

"message": "Query failure",

"data": "Error when querying chaincode:   
 Error:Failed to execute transaction or   
 query(Failed to retrieve V5C)"

},

"id": <id>

}

Reason:

One of the vehicle's that has a key stored in the index is   
 corrupt or missing.

# Glossary

api\_url

The URL of the peer and its API port that the chaincode should use to call out to.

caller

The identity of the user on the Blockchain who made the call.

chaincode\_name

The id of the chaincode. It is returned when chaincode is deployed and is used to tell the API which chaincode you wish to invoke or query. It is a 128 character string.

chaincode\_path

The link to where the chaincode is located either on the local file system or on GitHub. The link is not to the actual file but the folder containing the go file.

id

Chaincode requests via the HyperLedger Fabric API use an integer ID that will appear on the response so that a user can send multiple requests at once and know which response ties to which request. If the ID is 0 then the request is treated by fabric as a notification.

recipient

The identity of the user on the Blockchain who is receiving the vehicle.

transaction\_id

The ID of a transaction on the ledger. This uuid can be used with the HyperLedger API to select a transaction and view it.

username

The identity of a user on the Blockchain.

v5c\_ID

The unique identifier for a vehicle object. It is used as the key for the vehicles JSON object when it is written to the world state. It consists of 2 uppercase letters followed by 7 letters for example AB1234567.

vehicle\_json

The JSON object that defines a vehicle and is stored in the world state. The object is defined as:

{"v5cID": "<v5c\_ID>", "VIN": <VIN>, "make": "<string>", "model": "<string>", "reg": "<string>", "owner": <username>, "colour": "<string>", "leaseContractID": "<string>", "status": <int>, "scrapped": <bool>}

VIN

15 character integer or 0 if the vehicle has yet to be defined.