

# CA Service Virtualization 10.1- Test Drive

## General

The DevTest SV Test Drive is aimed to improve adoption of the DevTest product. It will provide the platform to introduce and to demonstrate the powerful features of the DevTest product suite to new users supported by

- Video based educational training
- Guided hands-on labs in a full featured DevTest installation
- References to product documentation
- Marketing collateral

DevTest SV Test Drive comes with a set of sample use cases. The user is guided through the use cases by hands-on lab using the DevTest Portal and a demo application.

DevTest SV Test Drive resides in a public cloud environment and provisioned by CA. DevTest will be licensed for a limited period.

DevTest SV Test Drive will be built upon DevTest 10.1. Demo application is Forward Cars in the same build.

This document covers the use cases for the hands-on labs.

## Test Drive Requirements

The use cases in Test drive

1. Provide Best User Experience
2. Is based on DevTest Portal as the single User Interface
3. Leverage the latest DevTest version (I.e.10.1)

## Prerequisites

Prior to walking the user through the SV Test Drive use cases, the user should be familiar with the DevTest environment and of the applications used throughout the demonstrations:

- DevTest architecture, components and their main tasks
  - Virtual Service Environment (VSE) for hosting virtual services
  - Coordinator to stage test cases to Simulators for execution
- Basic structure and usage of Portal
- Executing the demo application
- Download, Installation and Usage of Chrome extension
- Download of sample files required for use cases

To understand the use cases and the demonstration, following concepts are required to understand:

- Concept of default responses in Virtual Services
- Concept of execution modes in Virtual Services
- Concept of variables in test cases, configured by project configuration files or set during execution
- Differences between Baseline and Functional Tests

## Use Case Overview



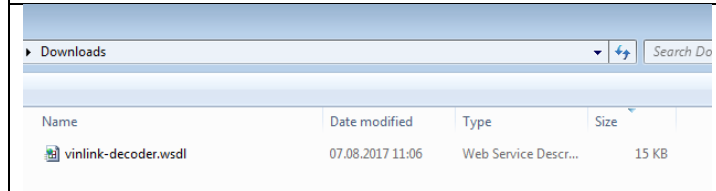
Following Use Cases are available in the DevTest Test Drive:

- 1. Use Case 1 – Demonstrate the usage of WSDL to create a virtual service**
  - a. Create VS from WSDL file
  - b. Execute Auto Generated Baseline Tests to run against virtual service
- 2. Use Case 2 – Demonstrate the usage of RR Pairs to create a virtual service**
  - a. Use RR pairs to create virtual service
  - b. Execute Auto Generated Baseline Test to run against virtual service
- 3. Use Case 3 – Demonstrate classic Web Service Recording and VS Execution Mode Change**
  - a. Classic recording of a sample web service to create a virtual service
  - b. Manual test of the virtual service using the sample web UI
- 4. Use Case 4 – Demonstrate Chrome Extension and Test Configuration Overriding**
  - a. Install and Configure Chrome Extension
  - b. Generate API Regression Test
  - c. Run API Regression Test against Real Service with generated Test Configuration
  - d. Run API Regression Test against Virtual Service by Overriding properties in generated Test Configuration

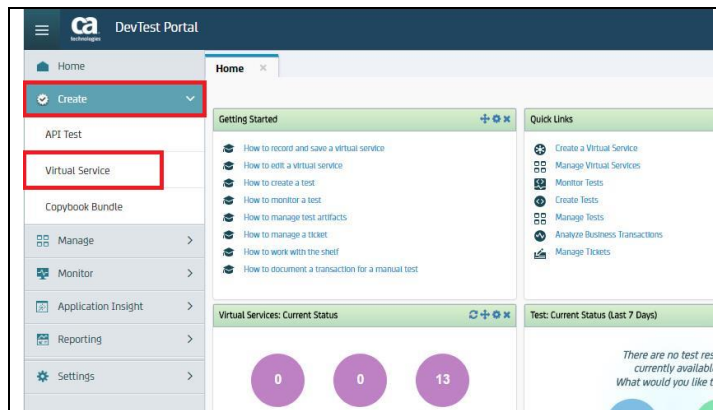
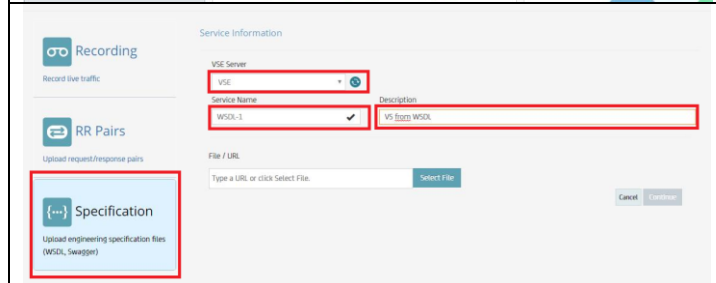
## Use Case 1 – VS from WSDL

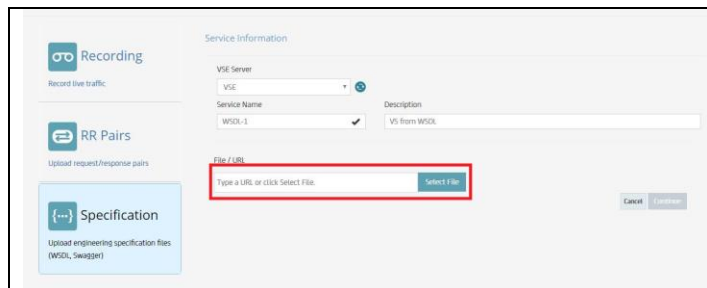
This use case requires a WSDL file. A sample WSDL file is provided with the SV Test Drive. It can be downloaded from SV Test Drive home page.

### Download WSDL Sample file

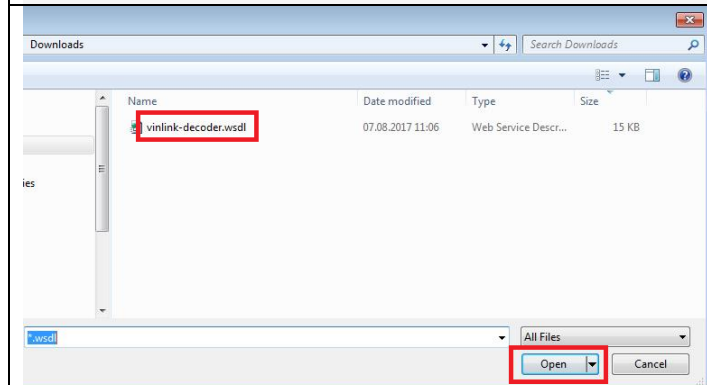
	<ol style="list-style-type: none"><li>1. Open SV Test Drive landing page</li><li>2. Hover the mouse of SV Asset button<ol style="list-style-type: none"><li>a. The Get Assets window pops up.</li></ol></li><li>3. Click on Get Assets button</li></ol>
	<ol style="list-style-type: none"><li>4. Click on 'Download vinlink-decoder.wsdl' to download the sample WSDL file</li></ol>
	<ol style="list-style-type: none"><li>5. The file appears in the local download folder</li></ol>

### Create Virtual Service

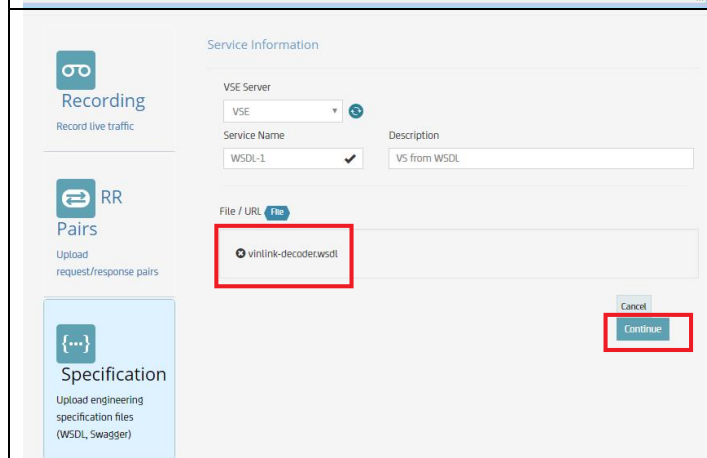
	<ol style="list-style-type: none"><li>6. Open DevTest Portal</li><li>7. Navigate to 'Create &gt; Virtual Service'</li></ol>
	<ol style="list-style-type: none"><li>8. In RHP, click on '{...} Specification'</li><li>9. Select a VSE from Drop down list</li><li>10. Enter a Virtual Service Name,</li><li>11. Optionally, enter a description</li></ol>



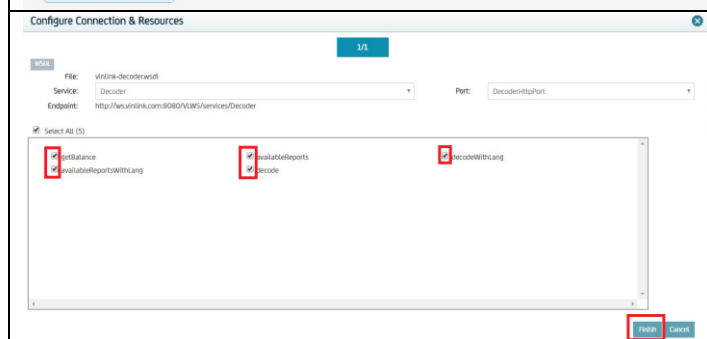
12. Click on 'Select File'
13. In File Explorer, navigate to LISA\_HOME/examples/data
14. Select 'vinlink-decoder.wsdl'
15. Click Open



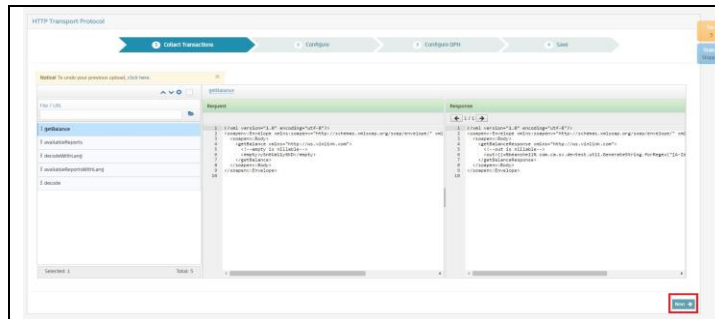
16. In File Explorer, navigate to Download folder, where the previously downloaded WSDL was stored.
17. Select 'vinlink-decoder.wsdl'
18. Click Open



19. Notice the file reference of the selected WSDL file
20. Click on 'Continue'



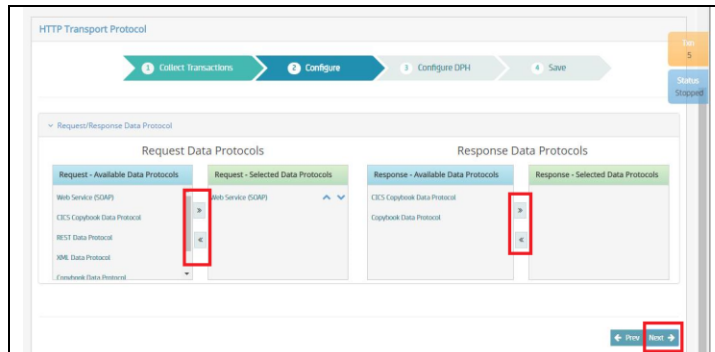
21. Review and select the operations for virtualization
22. Click 'Finish'



23. Review operations, request and response data

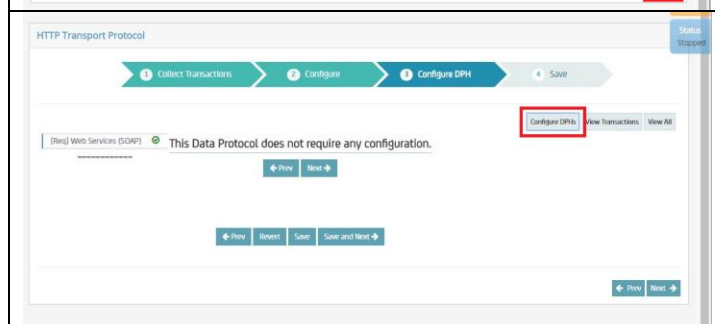
24. Click 'Next'

## Configure the Virtual Service

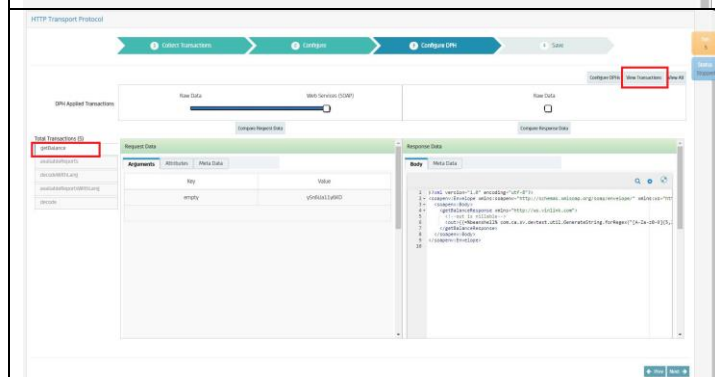


25. Review Data Protocol Handlers on both the Request and the Response Side

26. Click 'Next'

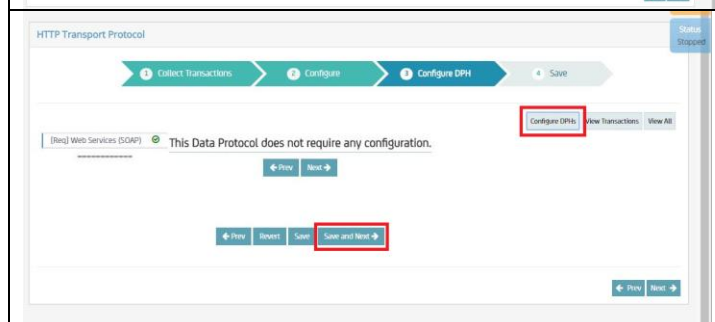


27. Click 'Configure DPHs' and notice that there is no need to configure the Web Service Data Protocol Handler



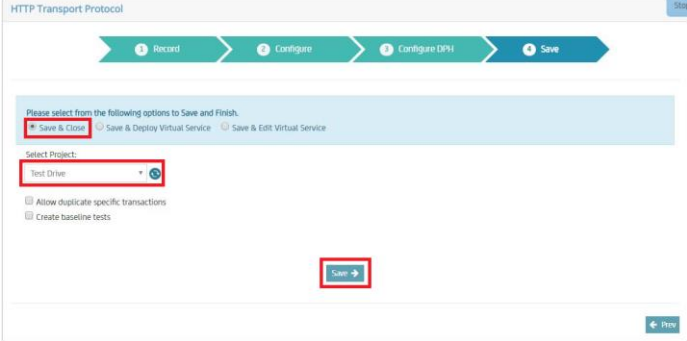

28. Click 'View Transactions' button and select a transaction name from list

29. Review the operations arguments, attributes and meta data, and the response data related to this request

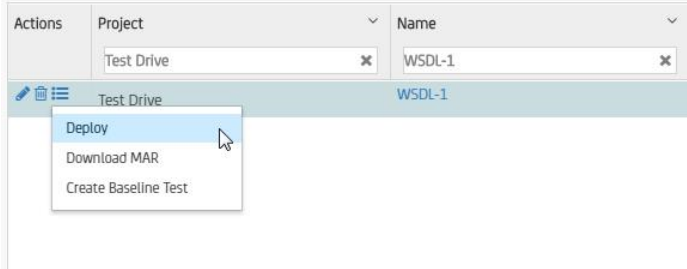
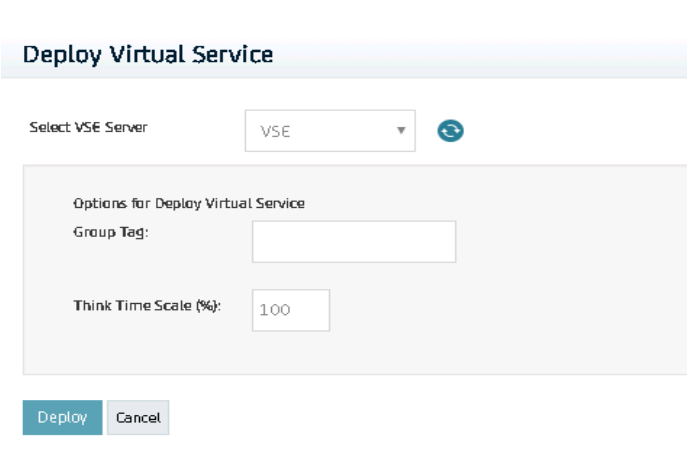
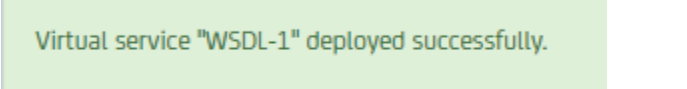


30. Click 'Configure DPHs'

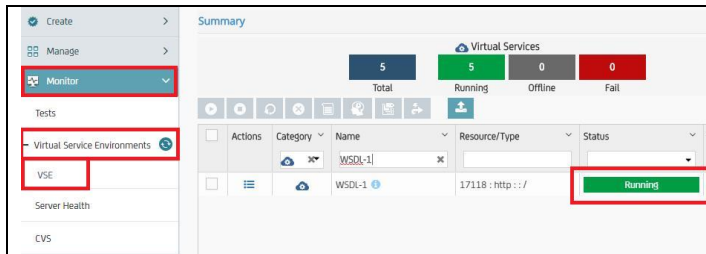
31. Click 'Save and Next' to conclude the configuration of the DPHs

	<p>32. Select 'Save &amp; Close' to save the virtual service and to close the wizard</p> <p>33. Select the project to save the virtual service in</p> <p>34. Click 'Save'</p>
	<p>35. Notice the name of the virtual service that was created and the project it was stored in.</p> <p>36. Confirm the message</p>

## Deploy Virtual Service

	<p>37. Navigate to 'Manage &gt; Virtual Services'</p> <p>38. From Project 'Test Drive' select the VS</p> <p>39. Click on the menu button and select 'Deploy'</p>
	<p>40. Review settings</p> <p>41. Click 'Deploy'</p>
	<p>42. Success message posted to top of pane</p>

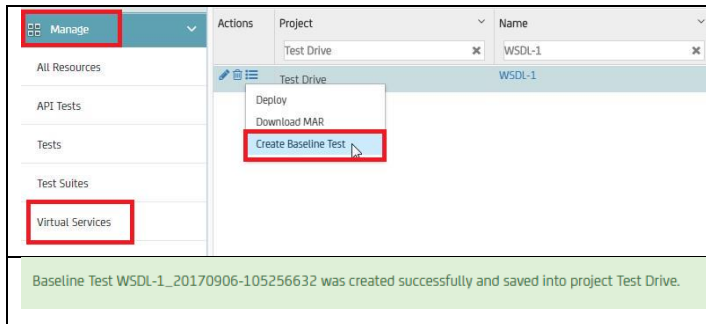
## Validate the Virtual Service is Running



43. Navigate to 'Monitoring > Virtual Service Environments > VSE'

44. Verify that VS is in 'Running' status

## Create Baseline Test from VS



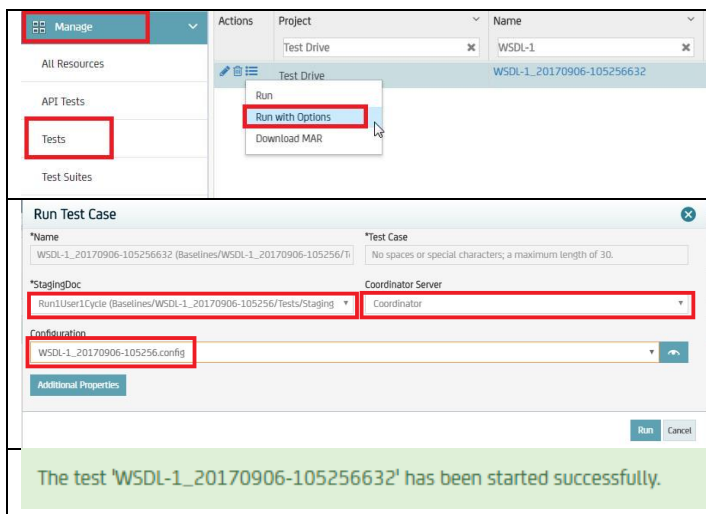
45. Navigate to 'Manage > Virtual Services'

46. Select VS

47. From Context menu select 'Create Baseline Test'

48. Success message posted on top of pane

## Launch Test Case



49. Navigate to 'Manage > Tests'

50. Select Test case

51. From context menu launch 'Run with Options'

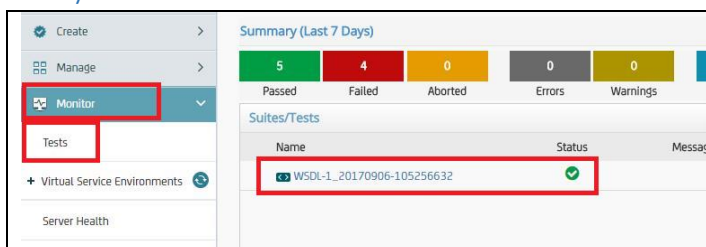
52. Review StagingDoc and Coordinator Server Settings

53. Select test specific project configuration file

54. Click 'Run'

55. Success message is posted on top of pane

## Verify Test Results



56. Navigate to 'Monitor > Tests'

57. Test case is launched

58. Once the test is completed, click on link

Overview > WSDL-1\_20170906-105256632

Status StagingDoc Documentation

1 0 0 0 0  
Passed Failed Aborted Errors Warnings

Cycle Details Cycle Num: 0 Status: Passed VUsers: 0 Start Time: 09/06/2017 10:57:03 AM Duration: 5ms

Work Flow Expand All Collapse All

Filter by 'Name' ☐ Steps ☒ Quiet Steps ☒ Fired Assertions ☒

- getBalance\_(virtualized)
- Assert Response Code Equals
- Assert Response Equals
- availableReports\_(virtualized)
- Assert Response Code Equals
- Assert Response Equals
- decodeWithLang\_(virtualized)

59. Review test steps and test results

availableReports\_(virtualized)

Request/Response

Request

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope"
3   xmlns:xs="http://www.w3.org/2001/XMLSchema">
4   <soapenv:Body>
5     <availableReports xmlns="http://ws.vinlink.com">
6       <!--vin is nillable-->
7       <vin>77P2GGy50</vin>
8       <!--language is nillable-->
9       <!--language supports enumerated values: en_US-->
10      <language>en_US</language>

```

Properties

Property	Value
lisa.dynamic.http.headers	<<null>>
lisa.ws.transport-headers.global.enab...	<<null>>
lisa.ws.transport-headers.service-dir...	<<null>>
lisa.ws.transport-headers.enabled	<<null>>

60. Expand any of the steps to drill into test step details

Home x Monitoring Tests x

Overview > WSDL-1\_20170906-105256632

Status StagingDoc Documentation

1 0 0 0 0  
Passed Failed Aborted Errors Warnings

61. You return to the test overview pane by clicking on the 'Overview' link in the details pane



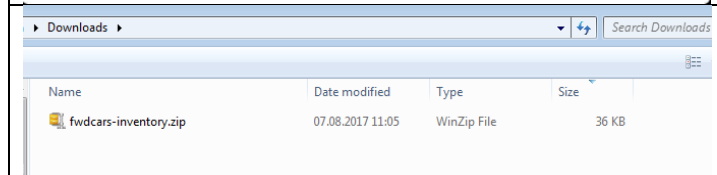


## Use Case 2 – VS from RR Pairs

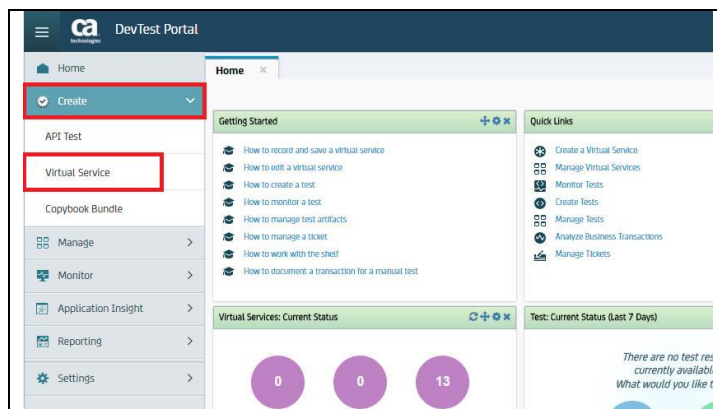
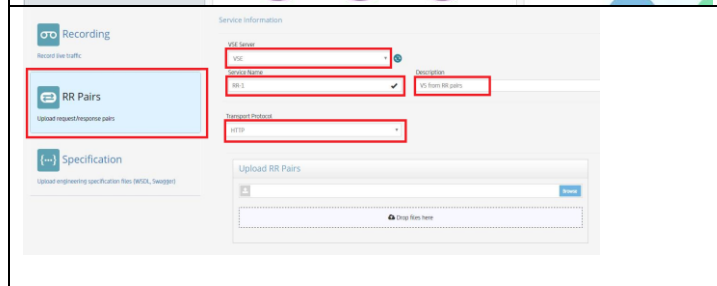
- RR pairs to create VS
- Execute Auto Generated API Tests run against VS

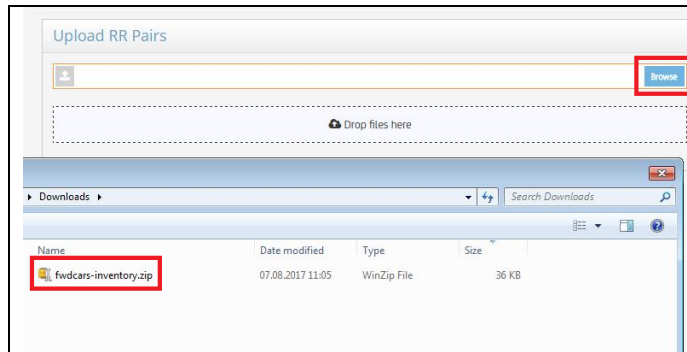
This use case requires a set of RR pairs. RR pairs can be loaded individually or as a set compressed by a zip file. A sample RR pair zip file is provided with the SV Test Drive. It can be downloaded from SV Test Drive home page.

### Download RR Pair Sample file

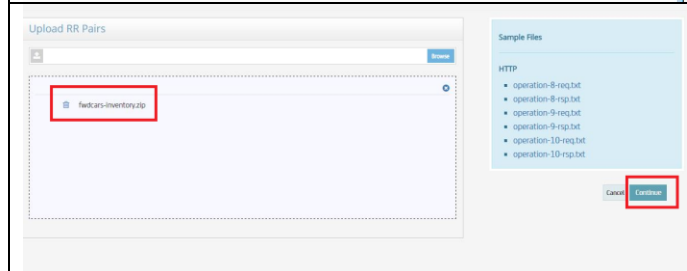
	62. Open SV Test Drive landing page 63. Hover the mouse of SV Asset button a. The Get Assets window pops up. 64. Click on Get Assets button
	65. Click on 'Download fwdcars-inventory.zip' to download the zip file containing a set of sample RR pairs.
	66. The file is stored on your local system in the download folder

### Create Virtual Service

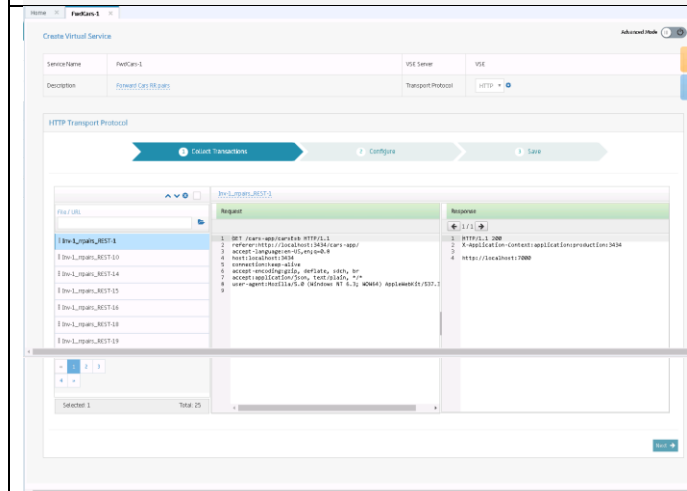
	67. Open DevTest Portal 68. Navigate to 'Create > Virtual Service'
	69. In RHP, click on 'RR Pairs' 70. Select a VSE from VSE Server drop down list 71. Enter a Virtual Service Name 72. Optionally enter a virtual service description 73. From Transport Protocol drop down list select 'HTTP'



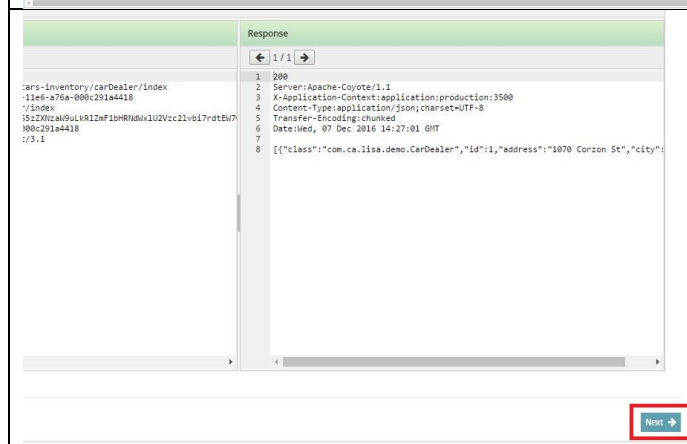
74. Click Browse to open the file explorer
75. Navigate to your local download folder
76. Select the downloaded zip file
77. Click open to upload the file



78. Note the filename of the uploaded file
79. Click 'Continue'

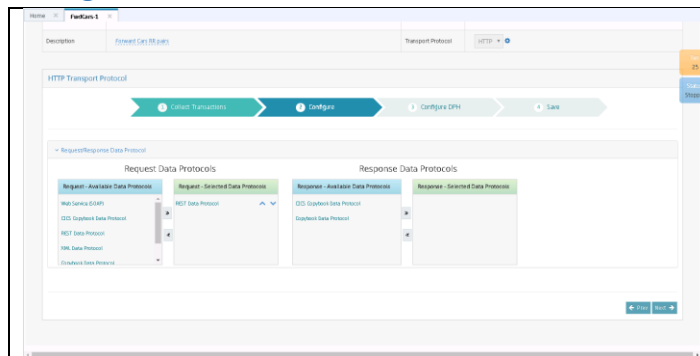


80. Review the imported RR pairs (requests/responses)

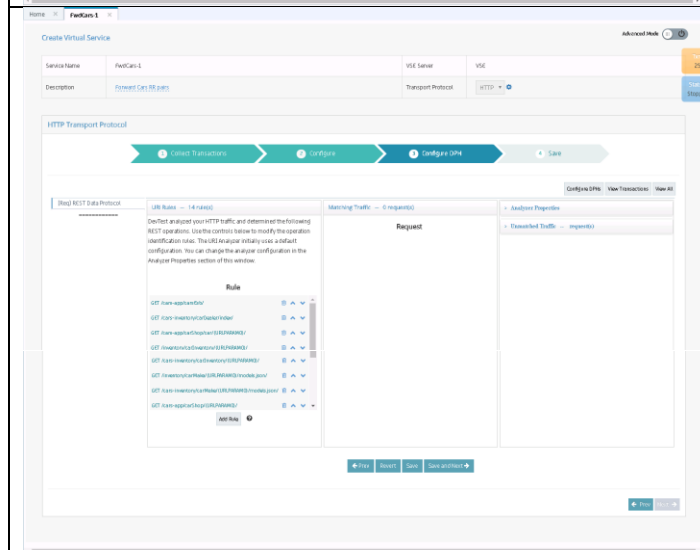


81. Scroll down and click next

## Configure the Virtual Service

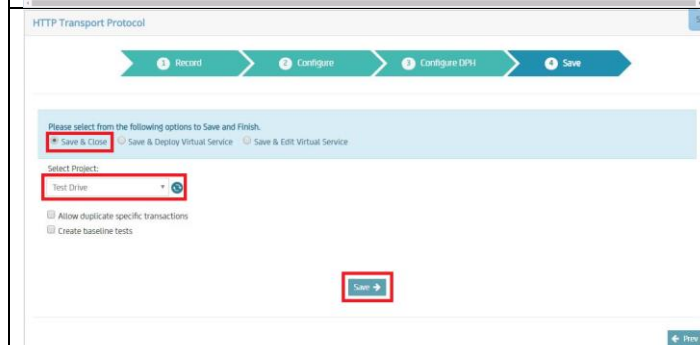


82. Select the DPHs required for the virtual service



83. Review and modify the Operation Identification rules to adjust Request/Response Matching

84. Click Save and Next to continue



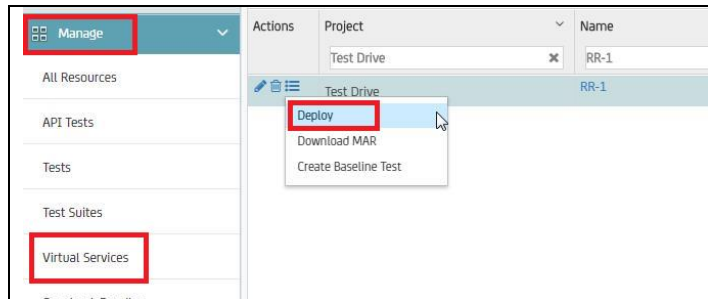
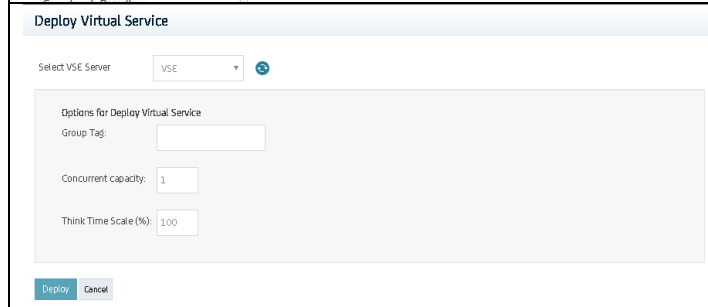
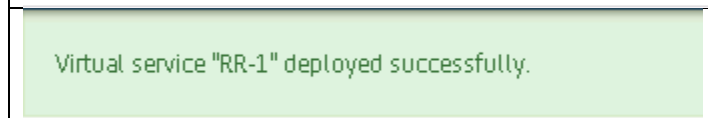
85. Select 'Save and Close'

86. Click Save to create the VS




87. Confirm the success message

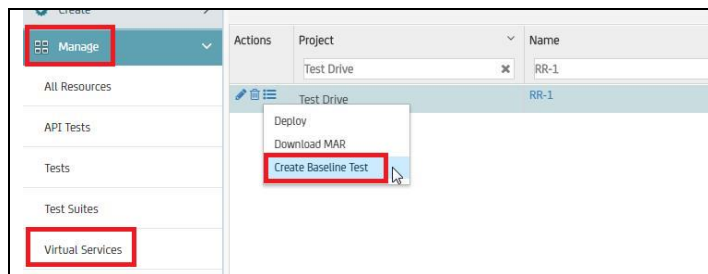

## Deploy Virtual Service

	<p>88. Navigate to 'Manage &gt; Virtual Services'</p> <p>89. Select the virtual service</p> <p>90. From context menu select 'Deploy'</p>
	<p>91. Optionally modify the settings</p> <p>92. Click Deploy to continue</p>
	<p>93. Note the success message at the top of the screen.</p>

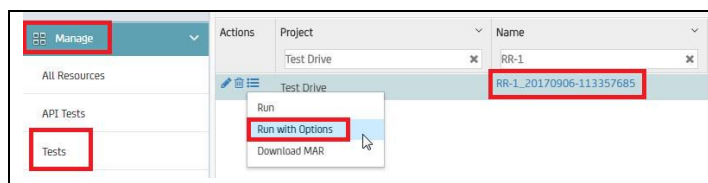
## Validate the Virtual Service is Running

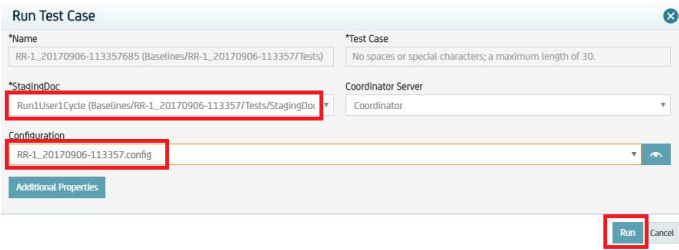
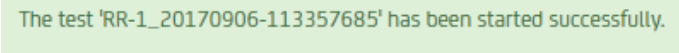
	<p>94. Navigate to 'Monitor &gt; Virtual Service Environments &gt; VSE'</p> <p>95. Verify VS is deployed, running and no error have occurred</p>
---	--

## Create Baseline Test from VS

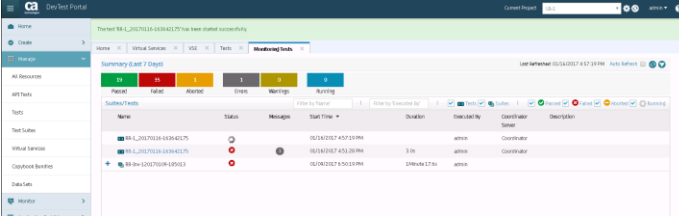
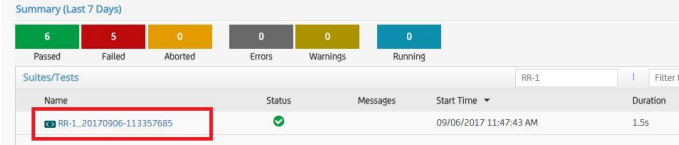
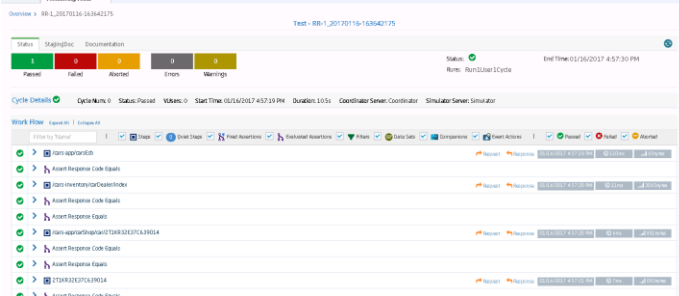
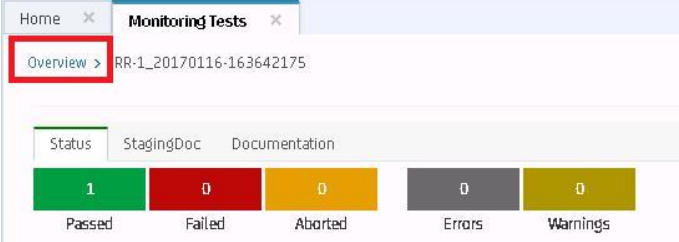
	<p>96. Navigate to 'Manage &gt; Virtual Services'</p> <p>97. Select created VS</p> <p>98. From Context menu select 'Create Baseline Test'</p>
	<p>99. Note the success message at the top of the screen.</p> <p>100. Take note of the Baseline test name</p>

## Launch Test Case

	<p>101. Navigate to 'Manage &gt; Tests'</p> <p>102. Select Baseline test name that was created previously</p> <p>103. From context menu select 'Run with options'</p>
---	---

	<p>104. Select the Staging doc related to this test case</p> <p>105. From Configuration list box select the project configuration file that was generated for this baseline test.</p>
	<p>106. Note the success message at the top of the screen</p>

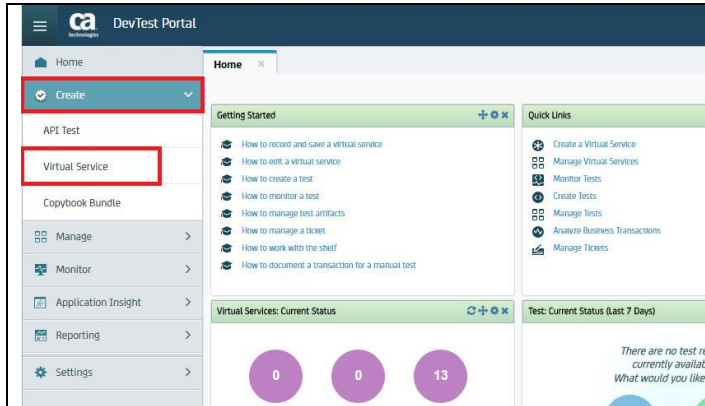
## Verify Test Results

	<p>107. DevTest Portal switches automatically to 'Monitoring &gt; Tests'</p>
	<p>108. After the test is finished, note the test result in the Status column</p> <p>109. Click on the test name to drill into the test result details</p>
	<p>110. The detailed results of the execution of each test step and of each assertion are displayed and can be validated</p>
	<p>111. You return to the test overview pane by clicking on the 'Overview' link in the details pane</p>

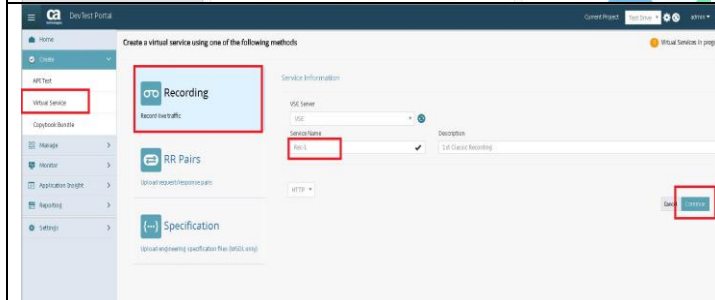
## Use Case 3 – VS from Recording

- Classic Recording
- Execute Auto Generated API Tests run against VS

### Record the Real Service



1. Open DevTest Portal
2. Navigate to 'Create > Virtual Service'






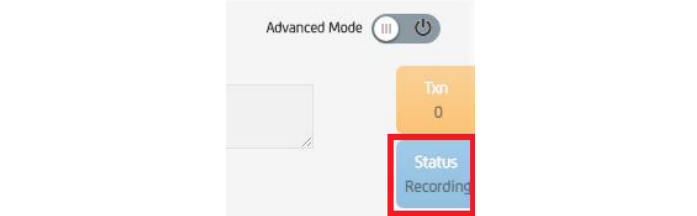
3. In RHP, click on 'Recording'
4. Select a VSE from list box
5. Enter a Virtual Service Name
6. Enter a Description of your recording
7. Select the 'HTTP' protocol
8. Click on 'Continue'



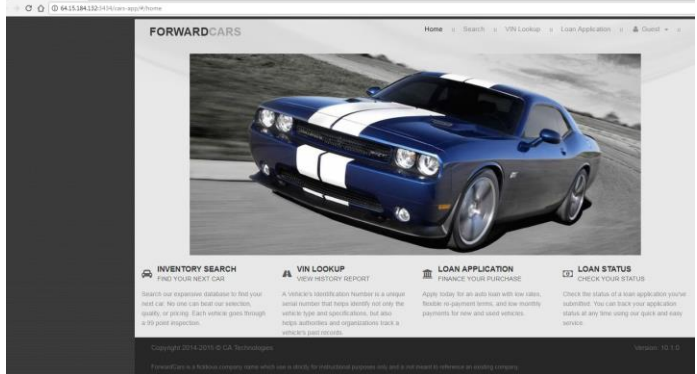
9. Enter URL to FwdCars demo application (<hostname>:3434)

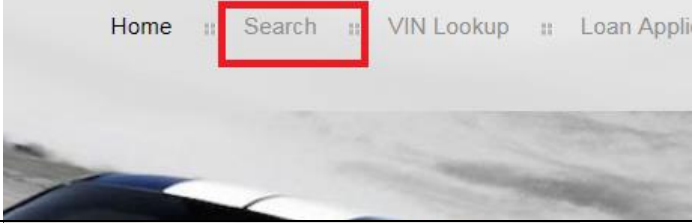
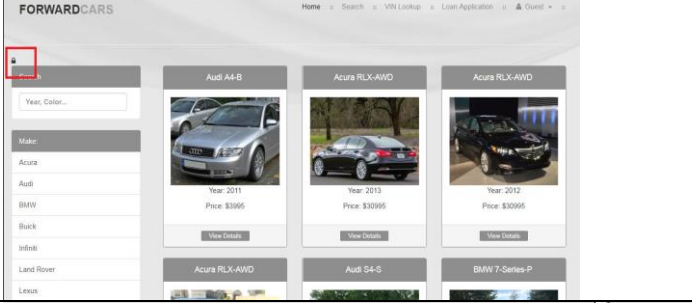
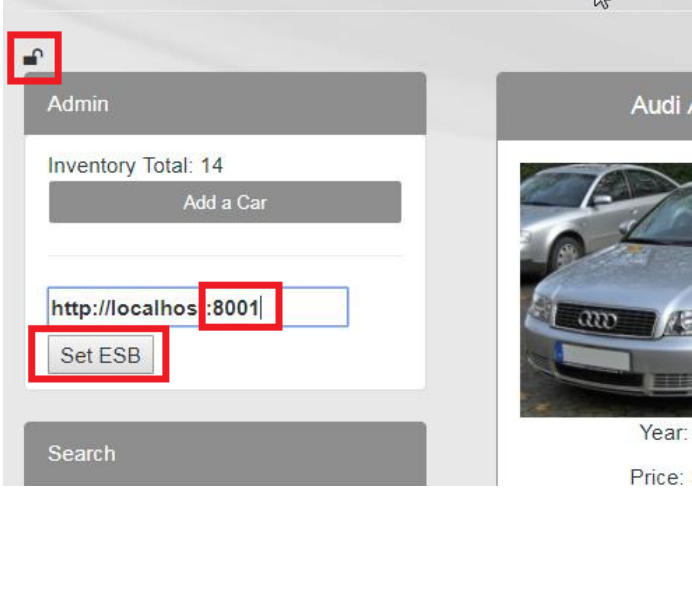
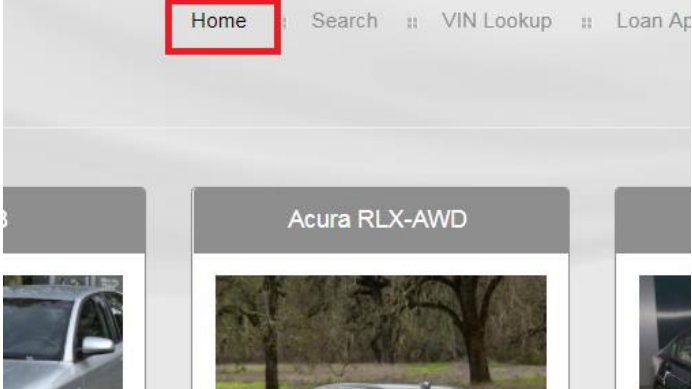


10. Click on Lock symbol to open it

 <p>The screenshot shows the VS Recorder interface. At the top, there's a diagram with a box labeled 'VS Recorder' containing a magnifying glass icon, connected by an arrow to a 'Server' icon. Below this, there's a dropdown menu set to 'All', a text input field containing '8001', and a lock icon. To the right of the lock icon is a checkbox labeled 'Use SSL' which is unchecked. Further right is a text field containing 'http://uv-dvtst10-1:34' with a green checkmark. At the bottom, there is a blue button labeled 'Start Recording'.</p>	<p>11. Enter a port number that is known to be open and available (i.e. not used by other applications or virtual services)</p> <p>12. Click the Lock symbol to close it again</p>
 <p>The screenshot shows the 'Advanced Mode' section of the VS Recorder interface. It features a power button icon and a 'Txn' counter showing '0'. Below these, there is a blue box with the text 'Status Ready'.</p>	<p>13. In the top right corner note the blue field with status 'Ready'</p>
 <p>This screenshot is identical to the first one, showing the VS Recorder interface with the 'Start Recording' button highlighted by a red rectangle.</p>	<p>14. Click Start Recording</p>
 <p>The screenshot shows the 'Advanced Mode' section of the VS Recorder interface. It features a power button icon and a 'Txn' counter showing '0'. Below these, there is a blue box with the text 'Status Recording'.</p>	<p>15. In the top right corner note the blue field with status 'Recording'</p>

## Set the ESB URL to VS Recorder

 <p>The screenshot shows the FORWARDCARS website. The header includes the logo and navigation links: Home, Search, VIN Lookup, Loan Application, and Loan Status. The main content area features a large image of a blue and white Ford Mustang. Below the image, there are four sections: 'INVENTORY SEARCH' (Find your next car), 'VIN LOOKUP' (View history report), 'LOAN APPLICATION' (Finance your purchase), and 'LOAN STATUS' (Check your status). The footer contains copyright information and a version number.</p>	<p>16. Open a new tab in browser</p> <p>17. Enter Recorder URL  <a href="http://&lt;srvr&gt;:3434/cars-app/#/home">http://&lt;srvr&gt;:3434/cars-app/#/home</a></p>
---	---

	<p>18. Click Search button in top row of the page</p>
	<p>19. Click the Lock icon at the right hand side of the page to open the ESB URL configuration dialog</p>
	<p>20. Configure the URL to point to the VS Recorder, i.e. enter the VS Recorder port number (8001 in the current example).</p> <p>21. Click Set ESB to save the new configuration.</p> <p>22. Click the Unlock icon to close the ESB URL configuration dialog</p> <p>Now the Web service points to the URL of the VS Recorder and will send any future ESB requests to the VS Recorder instead. The VS Recorder will record the request and forward it to the real service, receives its response, records it, too, correlates it to the request and forward the response to the client.</p>
	<p>23. Click Home to return to the home page of the application</p>



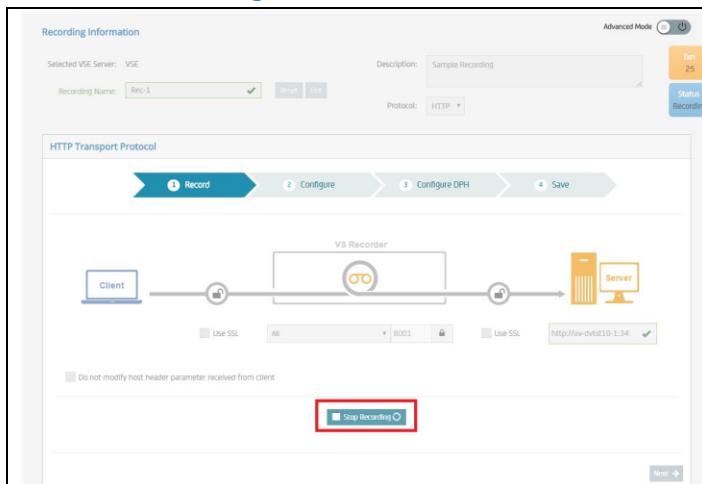
## Run the Web Application



24. From Home page run transactions for recording:

- Refresh the home page
- Use Inventory Search,
- see the details of cars
- lookup their VINs in the first row that is displayed.
- Click Home button to return to first screen

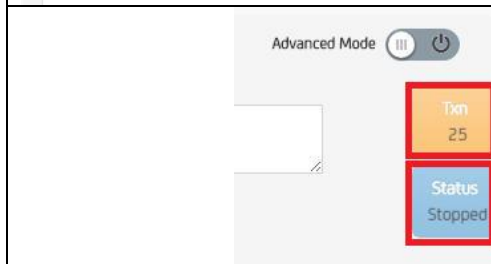
## Save the recording



25. Return to DevTest Portal

26. Click 'Stop Recording'

27.

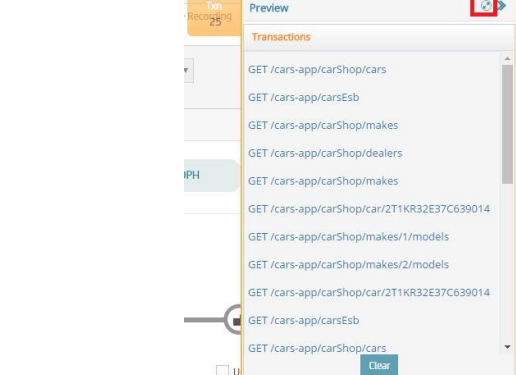
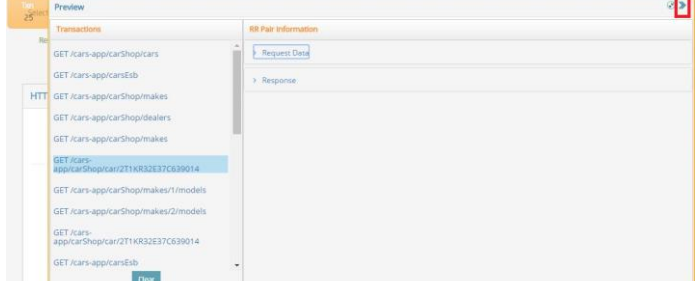



28. Review the Recorder status 'Stopped' in the blue colored field.

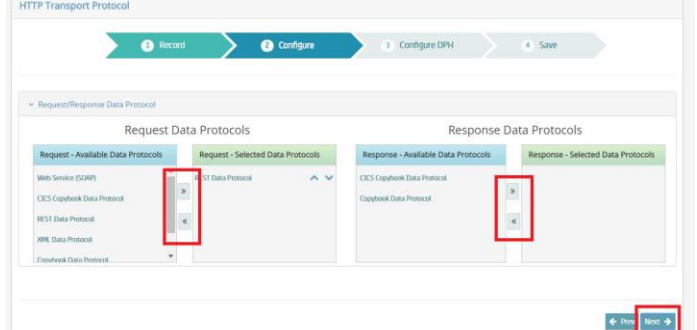
29. Review number of recorded transactions in the orange colored field.

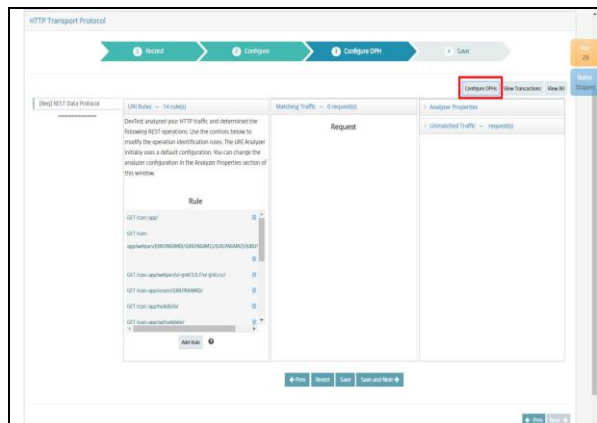
30. Click on the orange colored field to view the first 25 recorded transactions.

**Note:** There may be more recorded transactions, but only the first 25 are available and shown here.

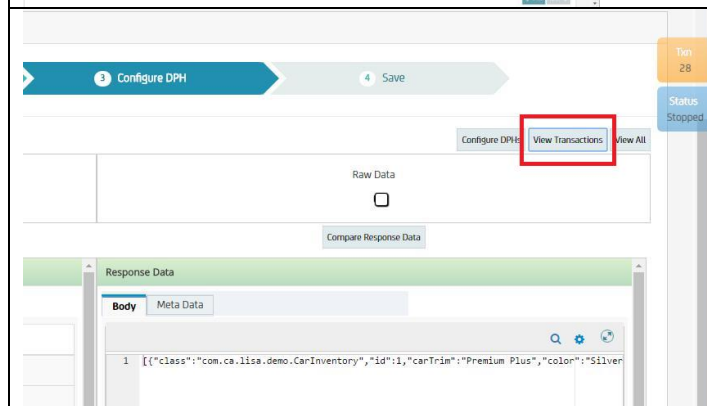
	<p>31. Click on the 'Expand' icon to open the detailed view of the recorded transactions</p>
	<p>32. Select a transaction from list in LHP  33. Expand 'Request Data' or 'Response' to view the details of the recorded request or response.  34. Click the 'Collapse' icon to close this view</p>
	<p>35. Click 'Next'</p>

## Configure the Virtual Service

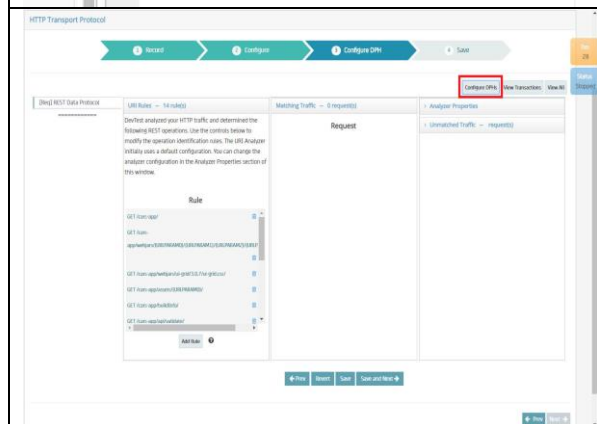
	<p>36. Review and accept or modify the recommended Data Protocol Handlers in 'Request – Selected Data Protocols' and 'Response – Selected Data Protocol Handlers'  37. Click 'Next'</p>
---	---



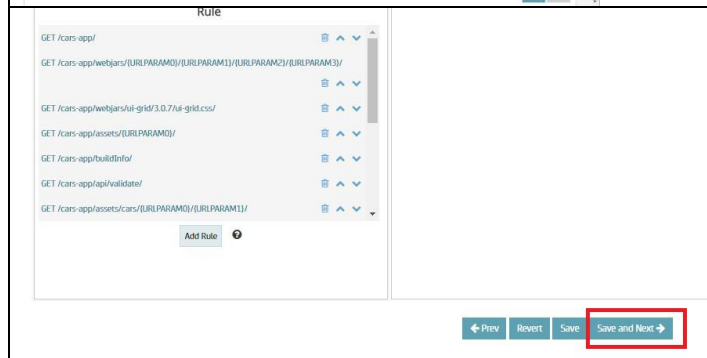
38. Click on 'Configure DPH' to modify the selected Data Protocol Handlers



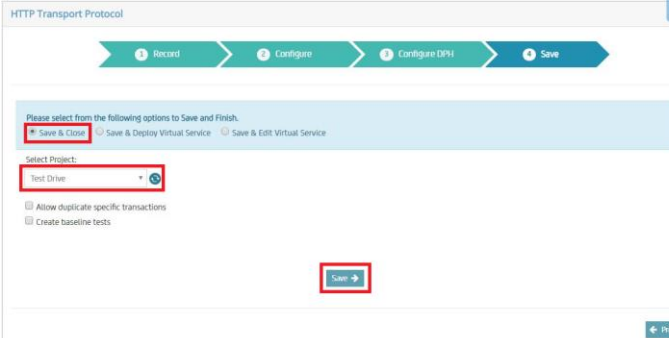
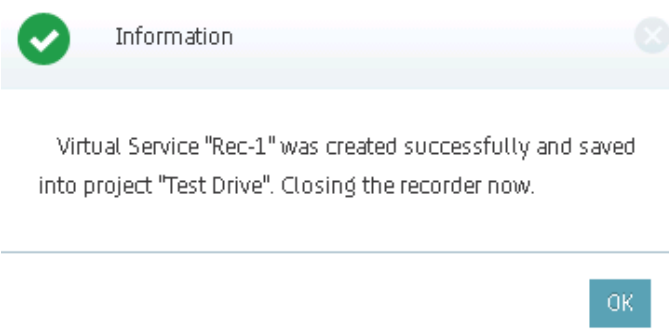
39. Click on 'View Transactions' to see Request and Response Data before (Raw) and after applying Data Protocol Handlers



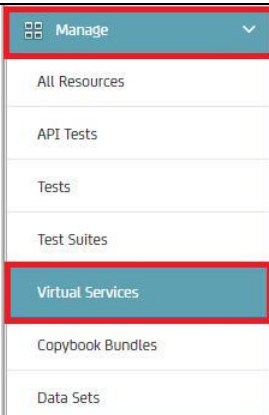
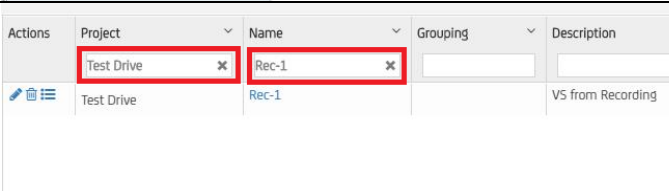
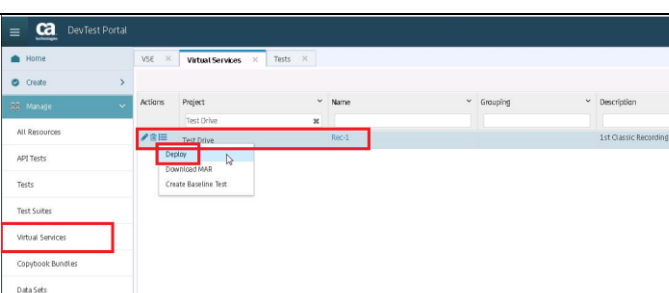
40. Click on 'Configure DPH' to return to Data Protocol Handler selection

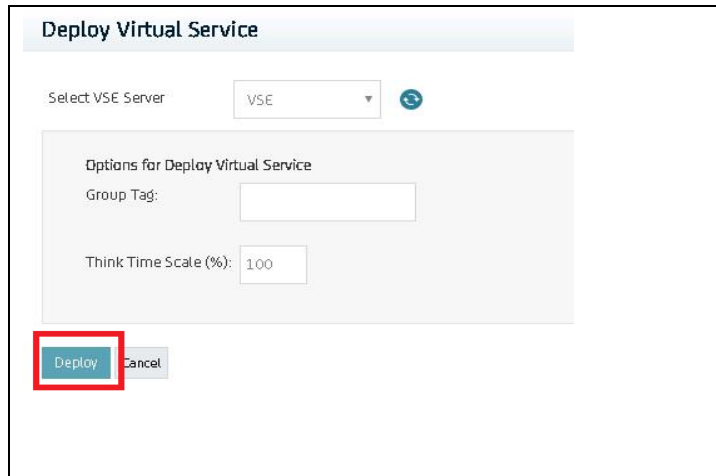
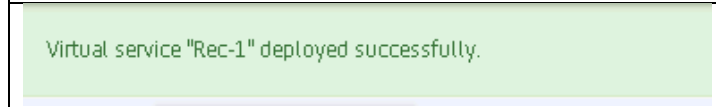


41. Click 'Save & Next'

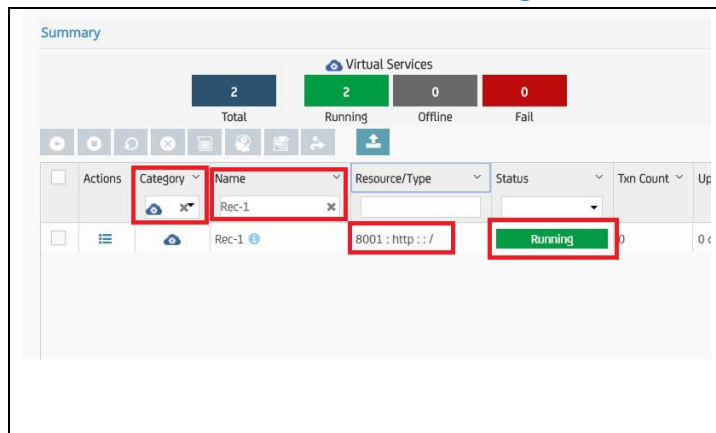
	<p>42. Select 'Save &amp; Close' to save the virtual service and to close the wizard</p> <p>43. Select the project to save the virtual service in</p> <p>44. Click 'Save'</p>
	<p>45. Confirm Success message</p>

## Deploy the Virtual Service

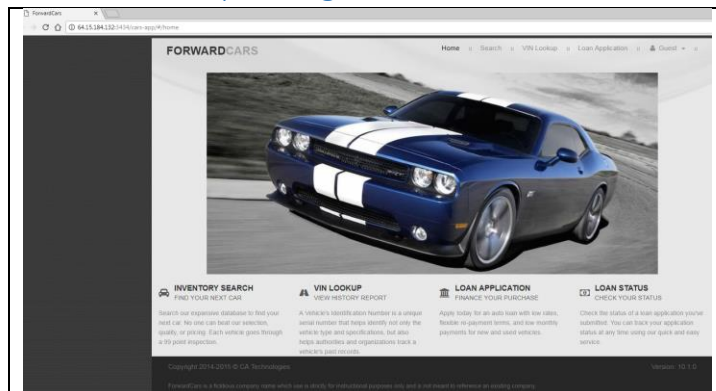
	<p>46. Navigate to 'Manage &gt; Virtual Services'</p>
	<p>47. To filter for the recorded virtual service,</p> <ol style="list-style-type: none"> <li>enter the project name in column 'Project'</li> <li>enter the virtual service name in column 'Name'</li> </ol>
	<p>48. From Project list select the row containing the virtual service name</p> <p>49. From context menu select 'Deploy'</p>

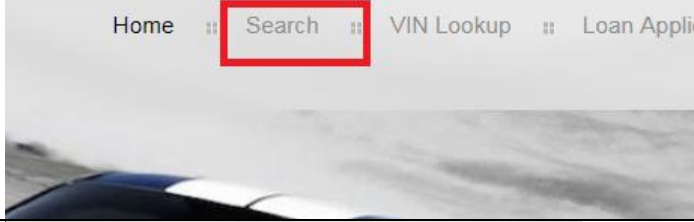
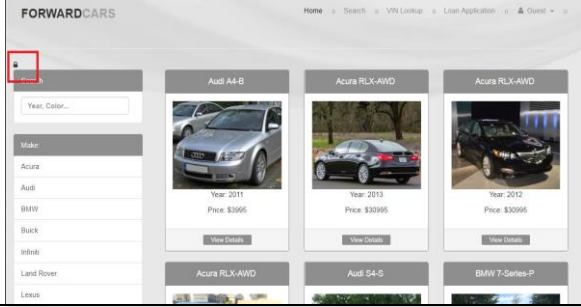
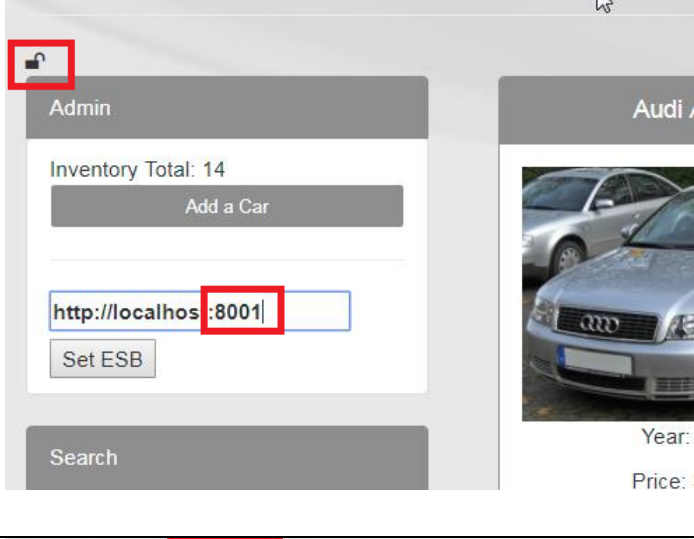
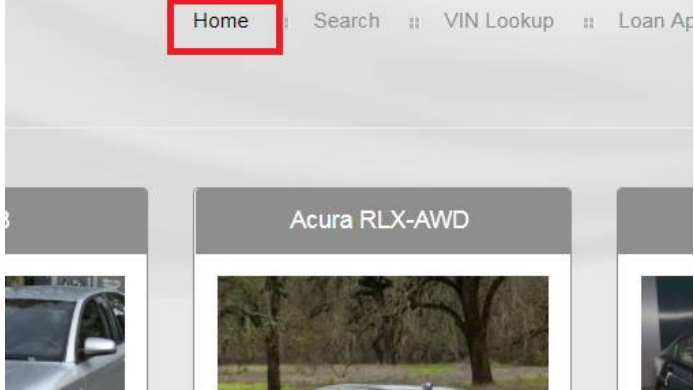
	<p>50. Review and change VSE Server to deploy the virtual service to</p> <p>51. Optionally, enter a tag for the virtual service to query for it</p> <p>52. Review and adjust 'Think Time Scale';</p> <ul style="list-style-type: none"> <li>• set Think Time Scale to 0% to enforce no response delay.</li> <li>• Leave it at 100% to have the same delay as when it was recorded.</li> <li>• Increase, if slower responses are required.</li> </ul> <p>53. Click 'Deploy'</p>
	<p>54. Notice success message on top of pane</p> <p><b>Take note of the virtual service name</b></p>

## Validate the Virtual Service is Running

	<p>55. Navigate to 'Monitor &gt; Virtual Service Environments &gt; VSE'</p> <p>56. In filter 'Category', select 'Virtual Service'</p> <p>57. In filter 'Name' enter the name of the virtual service</p> <p>58. Review 'Resource/Type', <b>The virtual service now listens on the port number previously used by the VS recorder</b></p> <p>59. Review 'Status', should be 'Running'</p> <p>60. Review 'Txn Count', should be 0</p>
--	--

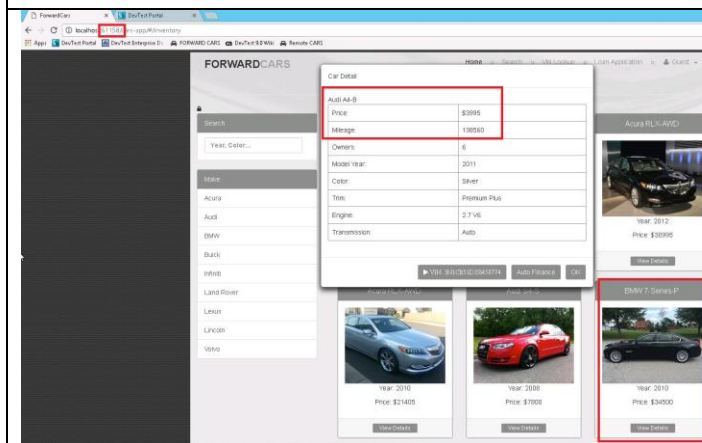
## Check ESB URL pointing to virtual service

	<p>61. Open a new tab in browser</p> <p>62. Enter Recorder URL <a href="http://localhost:3434/cars-app/#/home">http://localhost:3434/cars-app/#/home</a></p>
---	--

	<p>63. Click Search button in top row of the page</p>
	<p>64. Click the Lock icon at the right hand side of the page to open the ESB URL configuration dialog</p>
	<p>65. Check that the URL is still pointing to the port we set initially, i.e. this time pointing to the virtual service.</p> <p>66. Click the Unlock icon to close the ESB URL configuration dialog</p> <p>Now the Web service points to the URL of the virtual service running on the recorder port. The virtual service will try and match any future ESB requests to recorded requests. If a matching request is found the correlating response is returned. If the request matches, but the request parameter do not match, a default response is returned to the client instead.</p>
	<p>67. Click Home to return to the home page of the application</p>

## Test against Virtual Service

- Playback against the VS



68. Open a new tab in browser
69. Enter Recorder URL  
<http://<srvr>:3434/cars-app/#/home>

70. Run transactions for recording:  
**Note: Data are returned from virtual ESB service!**
  - a. Refresh the home page
  - b. Use Inventory Search,
  - c. see the details of cars
  - d. lookup their VINs in the first row that is displayed.
  - e. Click Home button to return to first screen

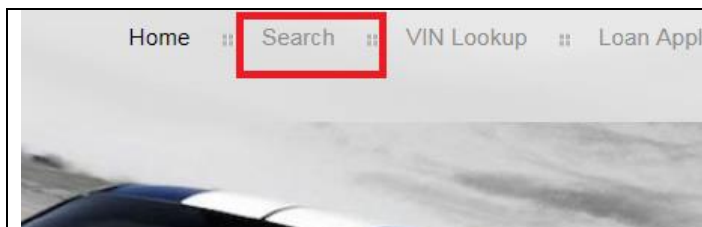
71. Review the data. They are the same as if they were returned by the live service.

72. Select a car that has not been recorded (first car in second row, for instance),

73. Review the returned data, which do not match the data of the selected car, but those of the first car recorded.

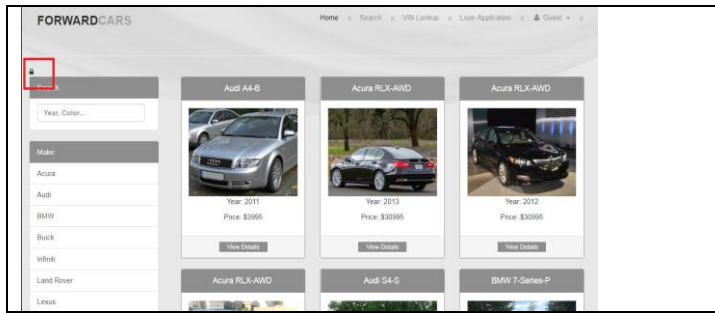
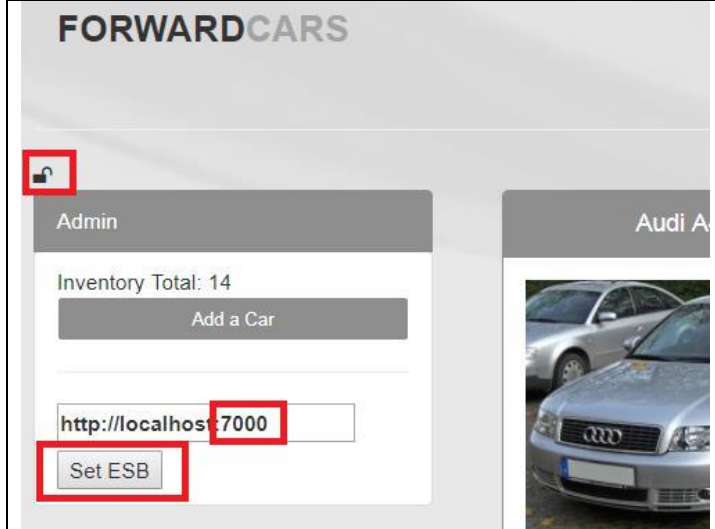
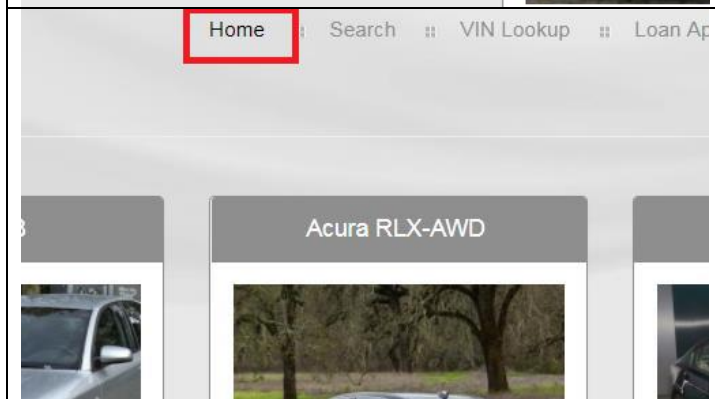
**Note: Response data of the first recorded car are returned by the virtual service as default responses for known requests with unmatched data**

## Set the ESB URL back to live service

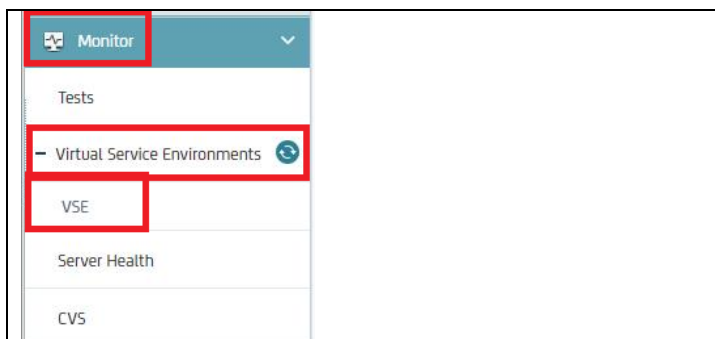


74. Click Search button in top row of the page



	<p>75. Click the Lock icon at the right-hand side of the page to open the ESB URL configuration dialog</p>
	<p>76. Configure the URL to point to the live ESB service again, i.e. enter the ESB URL with port number 7000.</p> <p>77. Click Set ESB to save the configuration.</p> <p>78. Click the Unlock icon to close the ESB URL configuration dialog</p> <p>Now the Web service points to the URL of the live ESB again.</p>
	<p>79. Click Home to return to the home page of the application</p>

Validate that the Virtual service was contacted

	<p>80. In DevTest Portal, navigate to 'Monitor &gt; Virtual Service Environments &gt; VSE'</p>
---	--



Summary

Last Refreshed: 09/06/2017 10:18:20 AM Auto Refresh 1.0

Virtual Services

Total: 4

Running: 4

Offline: 0

Fail: 0

Recordings

Total: 4

Recording: 0

Offline: 4

Fail: 0

Actions

Category

Name

Resource/Type

Status

Txn Count

Up-4%

Errors

Group

Exec:

Capacity

Task Scale

Auto-Resta...

Rec-1

Rec-1

8001 : http : /

Running

28

0 day...

M...

1

100

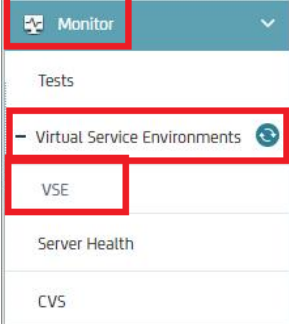
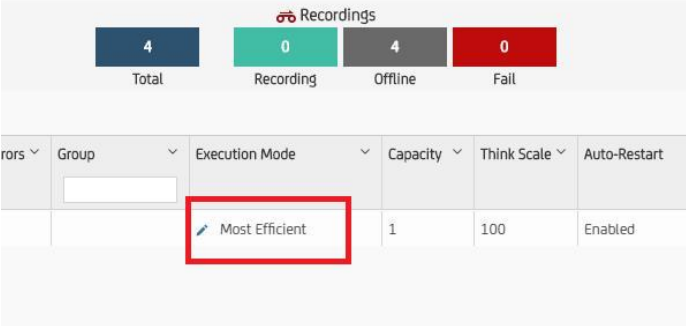
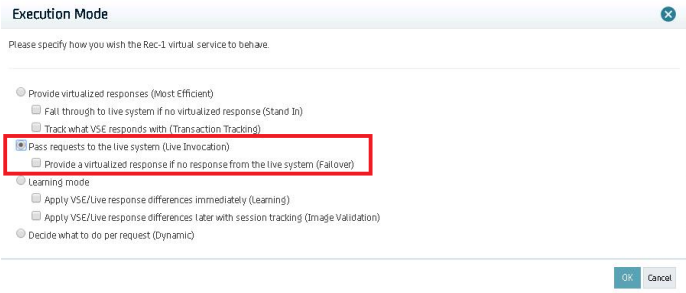
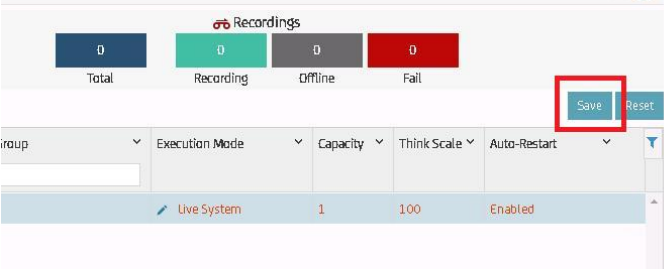
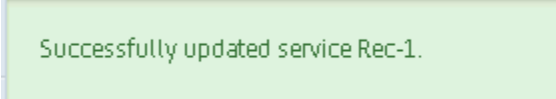
Enabled

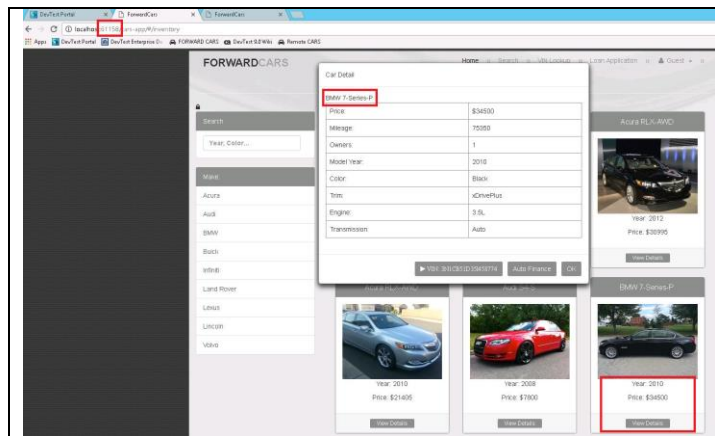
81. Review 'Txn Count', should be greater than 0 now, meaning that the web UI has hit the VS

## Use Case 3b - Test against Real Service

- Execution mode change
- Playback against the real service

### Change VS Execution Mode

	82. In DevTest Portal, navigate to 'Monitor > Virtual Service Environments > VSE'
	83. Review 'Resource Type', should point to port of VS, 84. Review 'Execution Mode', should be 'Most Efficient', which indicates an active Replay mode 85. Click on 'Edit' icon in front of 'Most Efficient'
	86. Change Execution Mode to 'Pass requests to live system (Live Invocation) 87. Click 'OK'
	88. Click on 'Save' to activate the new setting
	89. Note the success message on top of pane



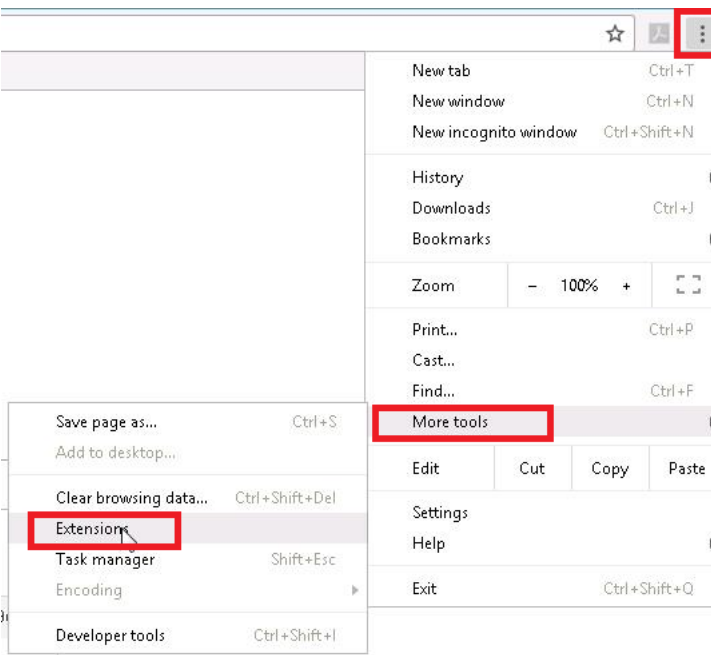

90. Validate Live Invocation usage (VS in pass thru mode) by running the Web UI against the VSE URL, including the VSE port number
91. Select a car that has not been recorded, and review the response from real system.

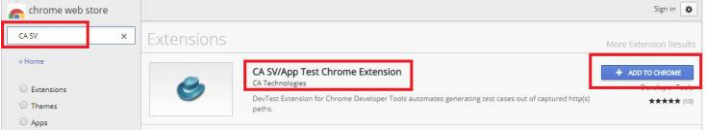
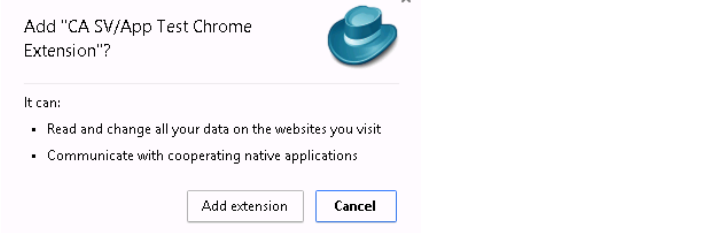
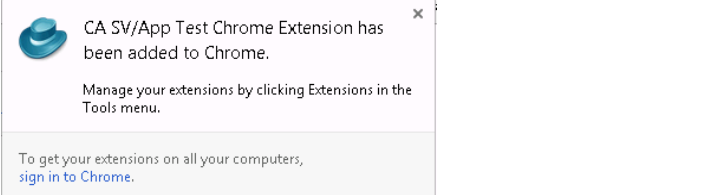
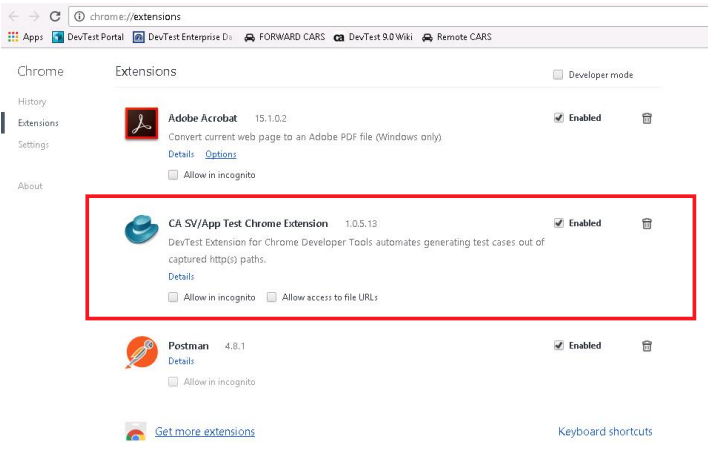
## Use Case 4 – DevTest Extension for Google Chrome

With the DevTest extension for Google Chrome browser the user can capture and analyze HTTP traffic and create functional tests from captured data. This feature is also referred to as the Chrome extension or the Chrome plug-in.

### Download and Install DevTest Extension for Google Chrome

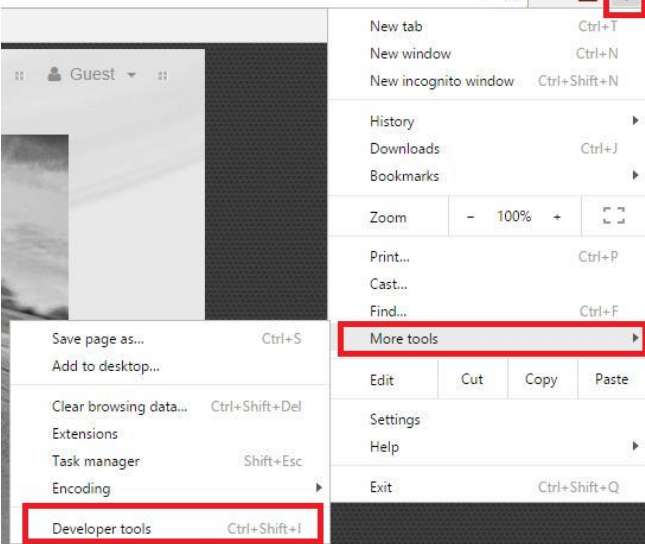
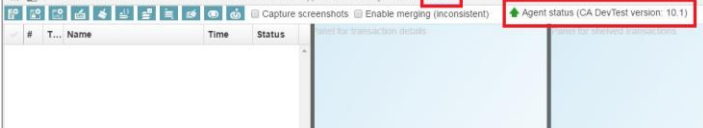
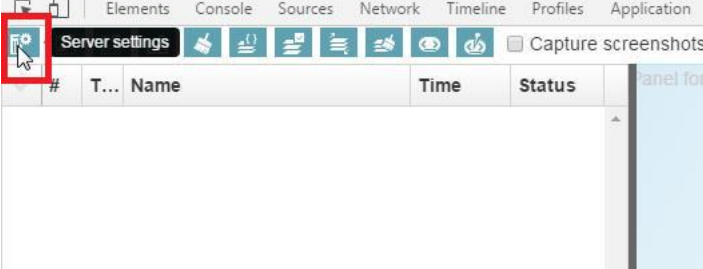

Installation of the DevTest Extension for Google Chrome is very easy and follows Chrome browser standards. It is available in Chrome webstore, and installation is familiar to users who have installed other Chrome extensions before.


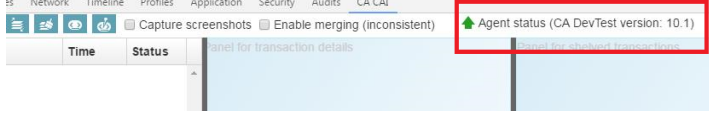
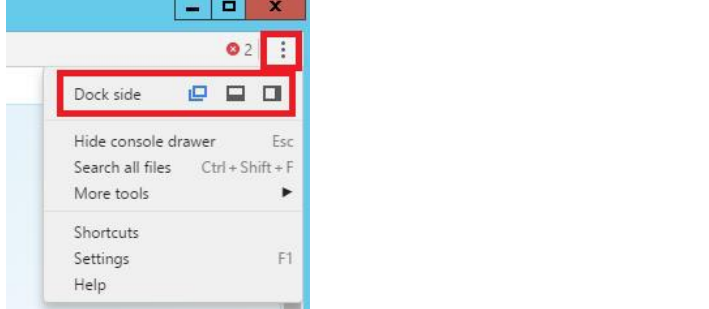
 <p>The screenshot shows the Google Chrome menu. The 'More tools' option is highlighted with a red box. Below it, the 'Extensions' option is also highlighted with a red box. Other visible options include 'New tab', 'New window', 'New incognito window', 'History', 'Downloads', 'Bookmarks', 'Zoom', 'Print...', 'Cast...', 'Find...', 'Save page as...', 'Add to desktop...', 'Clear browsing data...', 'Task manager', 'Encoding', and 'Developer tools'.</p>	<ol style="list-style-type: none"><li>1. In Google Chrome, navigate to 'Control and Customize Chrome' in the upper right corner,</li><li>2. Navigate to 'More tools &gt; Extensions'</li></ol>
 <p>The screenshot shows the Chrome Extensions page. The 'Get more extensions' button is highlighted with a red box. The page lists installed extensions: 'Adobe Acrobat' (version 15.1.0.2) and 'Postman' (version 4.8.1). Both are marked as 'Enabled'. The 'Developer mode' checkbox is also visible.</p>	<ol style="list-style-type: none"><li>3. Click on 'Get more extensions'</li></ol>

	<ol style="list-style-type: none"> <li>4. In chrome web store enter 'CA SV' in search field.</li> <li>5. When 'CA SV/AppTest Chrome Extension' is found, click on 'ADD TO CHROME' button.</li> </ol>
	<ol style="list-style-type: none"> <li>6. A dialog pops up to confirm the plugin installation.</li> <li>7. Click on 'Add Extension', after a few seconds</li> </ol>
	<ol style="list-style-type: none"> <li>8. Another dialog is displayed confirming the extension installation.</li> </ol>
	<p>In the list of Chrome extensions the CA SV/App Test Chrome extension is now listed, too.</p> <p>You can now use the extension to record transactions between the Chrome browser and the web service.</p>

## Configure DevTest Extension for Google Chrome

Now we want to use DevTest extension for Google Chrome to record Web service calls from local web client to the web service. We can launch the Chrome extension as soon as we are on the home or starting page of our user journey through the web application. So, we go back to the home page of our Forward Cars application.

 <p>A screenshot of the Google Chrome application menu. The 'More tools' option is highlighted with a red box. Below it, the 'Developer tools' option is also highlighted with a red box. Other visible options include 'New tab', 'New window', 'History', 'Downloads', 'Bookmarks', 'Zoom', 'Print...', 'Cast...', 'Find...', 'Save page as...', 'Add to desktop...', 'Clear browsing data...', 'Extensions', 'Task manager', 'Encoding', 'Edit', 'Cut', 'Copy', 'Paste', 'Settings', 'Help', and 'Exit'.</p>	<p>9. Once we are back on the starting page of our application, we need to launch the extension</p> <p>10. Select ,Control Chrome' &gt; More tools &gt; Developer tools</p>
 <p>A screenshot of the Chrome DevTools interface. The 'CA CAI' tab is selected in the top bar. Below it, the 'Agent status (CA DevTest version: 10.1)' is shown with a green status icon. The main panel displays a table with columns for '#', 'T...', 'Name', 'Time', and 'Status'.</p>	<p>11. Click on CA CAI</p> <p>12. Verify that Agent Status is up (green icon)</p>
 <p>A screenshot of the Chrome DevTools interface. The 'Server settings' tab is selected in the top bar. The main panel displays a table with columns for '#', 'T...', 'Name', 'Time', and 'Status'.</p>	<p>13. Click on Server settings (left most icon) to verify the location of the DevTest server to work with and to store the collected data.</p>
 <p>A screenshot of the 'Server Settings' dialog box. It contains fields for 'Host Name' (localhost), 'Console Port' (1505), 'Portal Port' (1507), 'User Name' (admin), and 'Password' (masked with dots). There are 'Update' and 'Cancel' buttons at the bottom.</p>	<p>14. These are default setting assuming a local standard DevTest installation.</p>

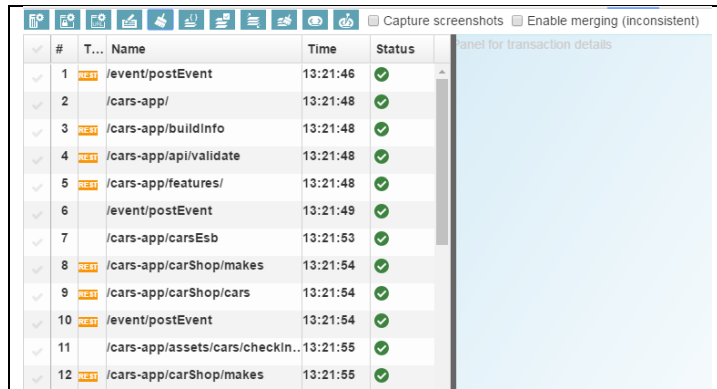
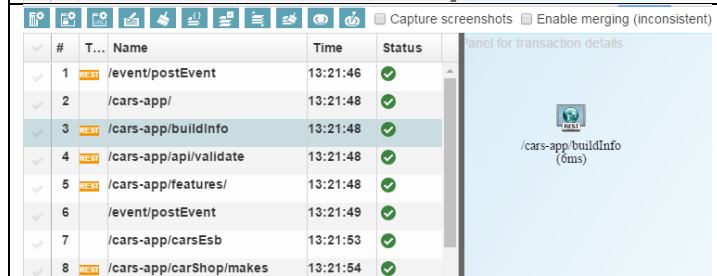
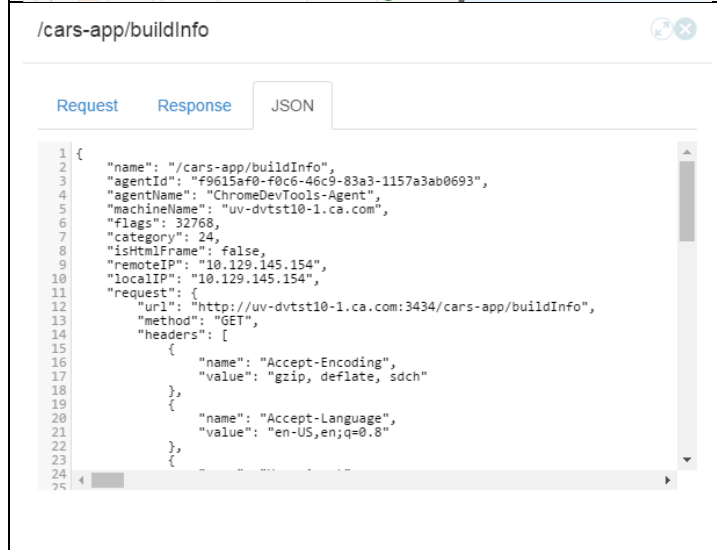
	<p>15. To access DevTest on the SV Test Drive server adjust the settings:</p> <ul style="list-style-type: none"> <li>• <b>Host Name or Host IP address</b> of the SV Test Drive Server</li> <li>• DevTest User Name: <b>devtest</b></li> <li>• DevTest User Password: <b>CAdemo123</b></li> </ul>
	<p>16. Verify the agent status is green. This means that the agent is connected to the DevTest server by the DevTest Broker.</p>
	<p>17. You can customize the location of the plugin display either to the bottom or to the right of your web browser, or you can undock it from it. Undocking is recommended.</p>

## Step 1 – Generate API Tests

### Generate API Tests from captured transactions

#### Record the User Journey and view the recorded Transaction Details

Now that we configured the extension to record the web service calls from our web client, we start the user journey through our application.

 <table border="1"><thead><tr><th>#</th><th>T...</th><th>Name</th><th>Time</th><th>Status</th></tr></thead><tbody><tr><td>1</td><td>REST</td><td>/event/postEvent</td><td>13:21:46</td><td>✓</td></tr><tr><td>2</td><td></td><td>/cars-app/</td><td>13:21:48</td><td>✓</td></tr><tr><td>3</td><td>REST</td><td>/cars-app/buildInfo</td><td>13:21:48</td><td>✓</td></tr><tr><td>4</td><td>REST</td><td>/cars-app/api/validate</td><td>13:21:48</td><td>✓</td></tr><tr><td>5</td><td>REST</td><td>/cars-app/features/</td><td>13:21:48</td><td>✓</td></tr><tr><td>6</td><td></td><td>/event/postEvent</td><td>13:21:49</td><td>✓</td></tr><tr><td>7</td><td></td><td>/cars-app/carsEsb</td><td>13:21:53</td><td>✓</td></tr><tr><td>8</td><td>REST</td><td>/cars-app/carShop/makes</td><td>13:21:54</td><td>✓</td></tr><tr><td>9</td><td>REST</td><td>/cars-app/carShop/cars</td><td>13:21:54</td><td>✓</td></tr><tr><td>10</td><td>REST</td><td>/event/postEvent</td><td>13:21:54</td><td>✓</td></tr><tr><td>11</td><td></td><td>/cars-app/assets/cars/checkin...</td><td>13:21:55</td><td>✓</td></tr><tr><td>12</td><td>REST</td><td>/cars-app/carShop/makes</td><td>13:21:55</td><td>✓</td></tr></tbody></table>	#	T...	Name	Time	Status	1	REST	/event/postEvent	13:21:46	✓	2		/cars-app/	13:21:48	✓	3	REST	/cars-app/buildInfo	13:21:48	✓	4	REST	/cars-app/api/validate	13:21:48	✓	5	REST	/cars-app/features/	13:21:48	✓	6		/event/postEvent	13:21:49	✓	7		/cars-app/carsEsb	13:21:53	✓	8	REST	/cars-app/carShop/makes	13:21:54	✓	9	REST	/cars-app/carShop/cars	13:21:54	✓	10	REST	/event/postEvent	13:21:54	✓	11		/cars-app/assets/cars/checkin...	13:21:55	✓	12	REST	/cars-app/carShop/makes	13:21:55	✓	<p>18. After browsing the web page, we can view the recorded web server calls</p> <ul style="list-style-type: none"><li>Recorded calls can be cleared by the broom icon (5<sup>th</sup> from left)</li></ul>
#	T...	Name	Time	Status																																																														
1	REST	/event/postEvent	13:21:46	✓																																																														
2		/cars-app/	13:21:48	✓																																																														
3	REST	/cars-app/buildInfo	13:21:48	✓																																																														
4	REST	/cars-app/api/validate	13:21:48	✓																																																														
5	REST	/cars-app/features/	13:21:48	✓																																																														
6		/event/postEvent	13:21:49	✓																																																														
7		/cars-app/carsEsb	13:21:53	✓																																																														
8	REST	/cars-app/carShop/makes	13:21:54	✓																																																														
9	REST	/cars-app/carShop/cars	13:21:54	✓																																																														
10	REST	/event/postEvent	13:21:54	✓																																																														
11		/cars-app/assets/cars/checkin...	13:21:55	✓																																																														
12	REST	/cars-app/carShop/makes	13:21:55	✓																																																														
 <table border="1"><thead><tr><th>#</th><th>T...</th><th>Name</th><th>Time</th><th>Status</th></tr></thead><tbody><tr><td>1</td><td>REST</td><td>/event/postEvent</td><td>13:21:46</td><td>✓</td></tr><tr><td>2</td><td></td><td>/cars-app/</td><td>13:21:48</td><td>✓</td></tr><tr><td>3</td><td>REST</td><td>/cars-app/buildInfo</td><td>13:21:48</td><td>✓</td></tr><tr><td>4</td><td>REST</td><td>/cars-app/api/validate</td><td>13:21:48</td><td>✓</td></tr><tr><td>5</td><td>REST</td><td>/cars-app/features/</td><td>13:21:48</td><td>✓</td></tr><tr><td>6</td><td></td><td>/event/postEvent</td><td>13:21:49</td><td>✓</td></tr><tr><td>7</td><td></td><td>/cars-app/carsEsb</td><td>13:21:53</td><td>✓</td></tr><tr><td>8</td><td>REST</td><td>/cars-app/carShop/makes</td><td>13:21:54</td><td>✓</td></tr></tbody></table>	#	T...	Name	Time	Status	1	REST	/event/postEvent	13:21:46	✓	2		/cars-app/	13:21:48	✓	3	REST	/cars-app/buildInfo	13:21:48	✓	4	REST	/cars-app/api/validate	13:21:48	✓	5	REST	/cars-app/features/	13:21:48	✓	6		/event/postEvent	13:21:49	✓	7		/cars-app/carsEsb	13:21:53	✓	8	REST	/cars-app/carShop/makes	13:21:54	✓	<p>19. We can view the details of each web service call by selecting in the left pane. It puts the icon into the middle 'Panel for transaction details'.</p> <p>20. Clicking on it opens</p>																				
#	T...	Name	Time	Status																																																														
1	REST	/event/postEvent	13:21:46	✓																																																														
2		/cars-app/	13:21:48	✓																																																														
3	REST	/cars-app/buildInfo	13:21:48	✓																																																														
4	REST	/cars-app/api/validate	13:21:48	✓																																																														
5	REST	/cars-app/features/	13:21:48	✓																																																														
6		/event/postEvent	13:21:49	✓																																																														
7		/cars-app/carsEsb	13:21:53	✓																																																														
8	REST	/cars-app/carShop/makes	13:21:54	✓																																																														
 <pre>1 { 2   "name": "/cars-app/buildInfo", 3   "agentId": "f9615af0-f0c6-46c9-83a3-1157a3ab0693", 4   "agentName": "ChromeDevTools-Agent", 5   "machineName": "uv-dvtst10-1.ca.com", 6   "flags": 32768, 7   "category": 24, 8   "isHtmlFrame": false, 9   "remoteIP": "10.129.145.154", 10  "localIP": "10.129.145.154", 11  "request": { 12    "url": "http://uv-dvtst10-1.ca.com:3434/cars-app/buildInfo", 13    "method": "GET", 14    "headers": [ 15      { 16        "name": "Accept-Encoding", 17        "value": "gzip, deflate, sdch" 18      }, 19      { 20        "name": "Accept-Language", 21        "value": "en-US,en;q=0.8" 22      }, 23      { 24        "name": "Accept-Charset", 25        "value": "utf-8;q=0.7,*;q=0.3" 26      } 27    ] 28  } 29 }</pre>	<p>21. The detail view of this web service call. It shows the request, the response data and, in this case the JSON formatted response.</p> <p>22. Close this view by clicking on the close (x) button in the top right corner</p>																																																																	







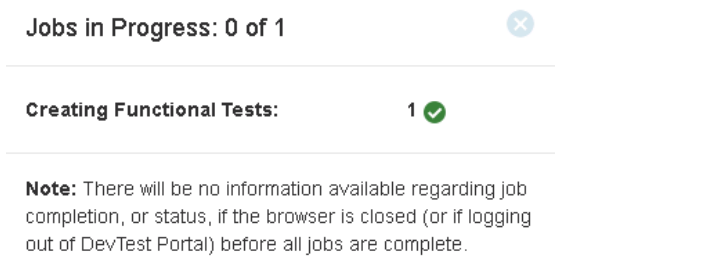
## Shelve Recorded API Calls

Creating a test case for regression testing from recorded transactions is two phase process. First, you select the web service calls that you want to test and copy them to a separate place that we call the 'shelf'. Once you have copied all the transactions to the shelf that you are interested in, you click a button that creates the test case.

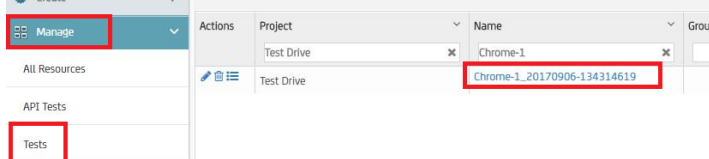
	<p>23. We have three ways to copy transactions to the shelf:</p> <ul style="list-style-type: none"><li>• select all the calls,</li><li>• select just the REST calls, or</li><li>• pick individual calls selectively.</li></ul> <p>If necessary, selected web service calls can be cleared from the shelf, again.</p>
	<p>24. We select to shelve all REST calls, but '/cars-app/api/validate'. This call checks the returned timestamp, which will be different, of course.</p> <p>25. Putting the selected web service calls to the shelf displays them in the 'panel for shelved transactions'. The user journey is visible by the order the calls are executed.</p>
	<p>26. Once shelved we can create a 'Functional Test' (2<sup>nd</sup> icon from right) or a 'Document Transaction object' plus 'Functional Test' (rightmost icon) from shelved frames.</p> <p>27. We shall create a Functional Test (2<sup>nd</sup> icon from right).</p>

## Create Functional Test

	<p>28. Clicking on the Functional Test icon opens this dialog.</p> <p>29. To create a functional test, often called a regression test, we opt for the Stateful mode.</p>
--	--

	<p>30. The test case will be created with a unique name. Here we specify the prefix for this name.</p> <p>31. We want to modify the name for better recognition.</p>
	<p>32. By clicking on the link, we change the name for something better to remember</p> <p>33. Confirm the change by clicking on the confirm icon</p>
	<p>34. Clicking on the create button opens the next dialog.</p>
	<p>35. Review the selected project</p> <p>36. Choose whether to keep or delete the items from shelf. It does not matter in this example.</p> <p>37. Click 'Create' to create the Functional Test</p>
	<p>38. Close the success message by clicking on the close (x) button in the top right corner.</p>

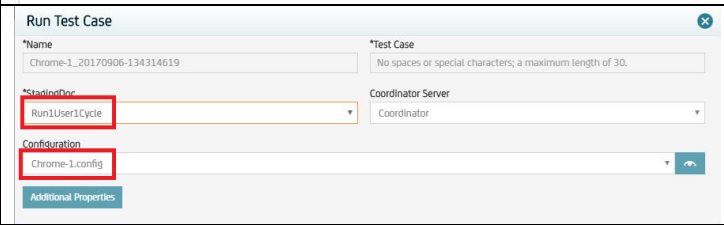
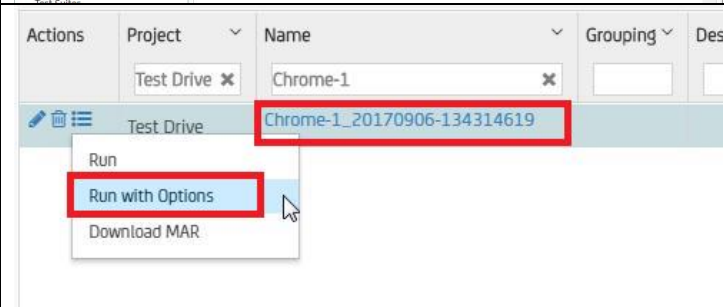
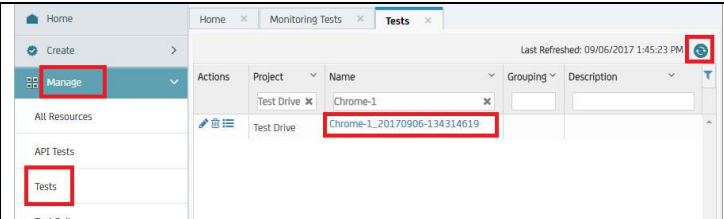
### Verify the Test Case was created

	<p>39. Navigate in Portal to 'Manage &gt; Tests'</p> <p>40. Click on 'Refresh' if the test case is not displayed</p>
---	--

Last Refreshed: 09/06/2017 1:45:23 PM

Step 2 – Test against Real Service

Execute the API Test Case against the real service



41. Navigate in Portal to ‘Manage > Tests’

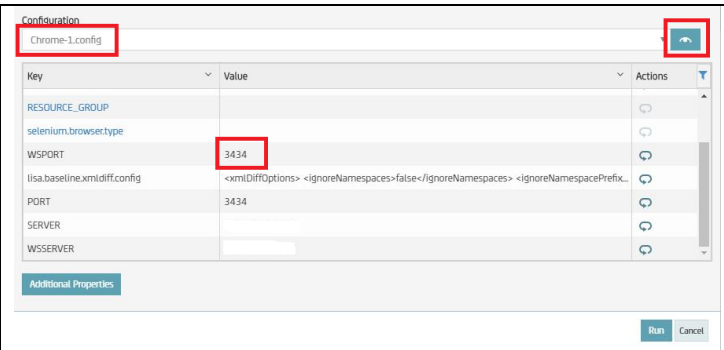
42. Click on ‘Refresh’ if the test case is not displayed

43. From context menu launch ‘Run with Options’

44. Review Staging Doc and Coordinator Server settings

45. Select project configuration file of generated functional test case

Review Current Test Configuration




46. Click on ‘Configuration Details’ icon (at the right end of the configuration file list box)

47. Scroll down to review key ‘WSPORT’, which should refer to the port of the real service on the local system.

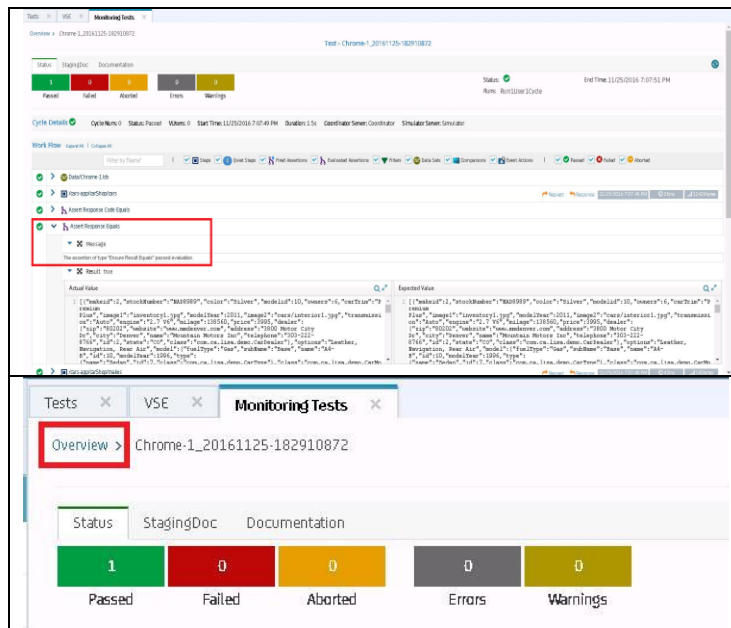
48. Click ‘Run’

Review Test Progress



49. Navigate to ‘Monitor > Tests’

50. Review the ‘Status’

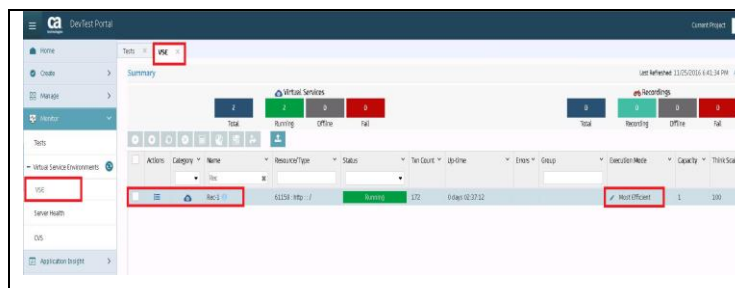


51. Once completed, drill down into test results by clicking on the test case link

52. Return to 'Monitor > Tests' overview by clicking on the 'Overview' link in the top left corner of the pane

### Step 3 – Test against Virtual Service

We override the default test configuration, which points to the real service, by changing variables, so that the test configuration now points to the virtual service.

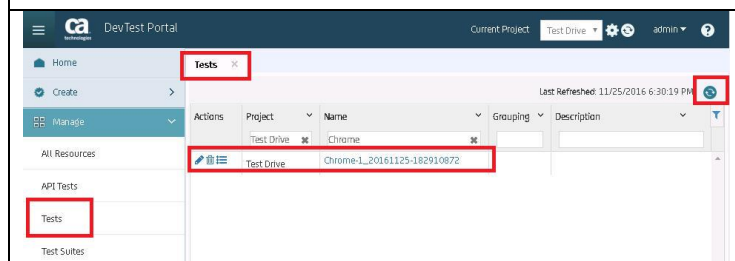


53. Navigate in Portal to 'Monitoring > Virtual Service Environments > VSE'

54. Review the port number the VS is listening on

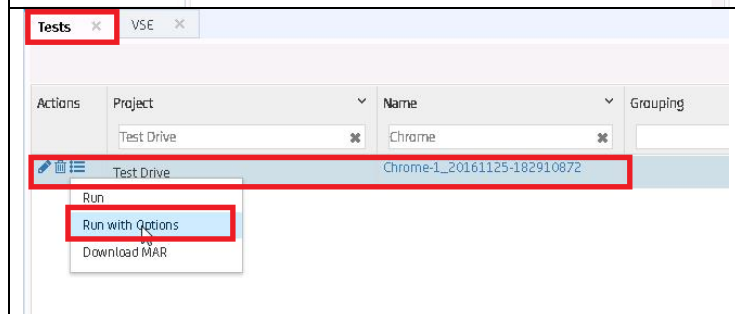
55. Review the status of the VS as being in 'Running'

56. Review the Execution mode being 'Most Efficient'

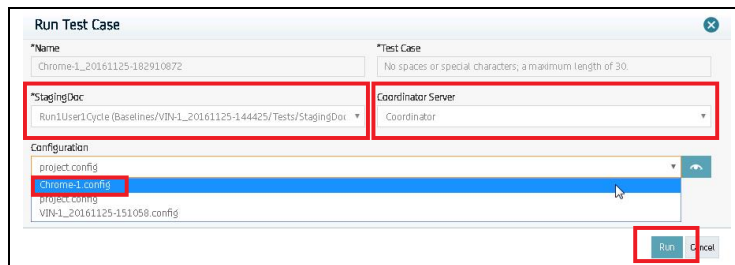


57. Navigate in Portal to 'Manage > Tests'

58. Click on 'Refresh' if the test case is not displayed

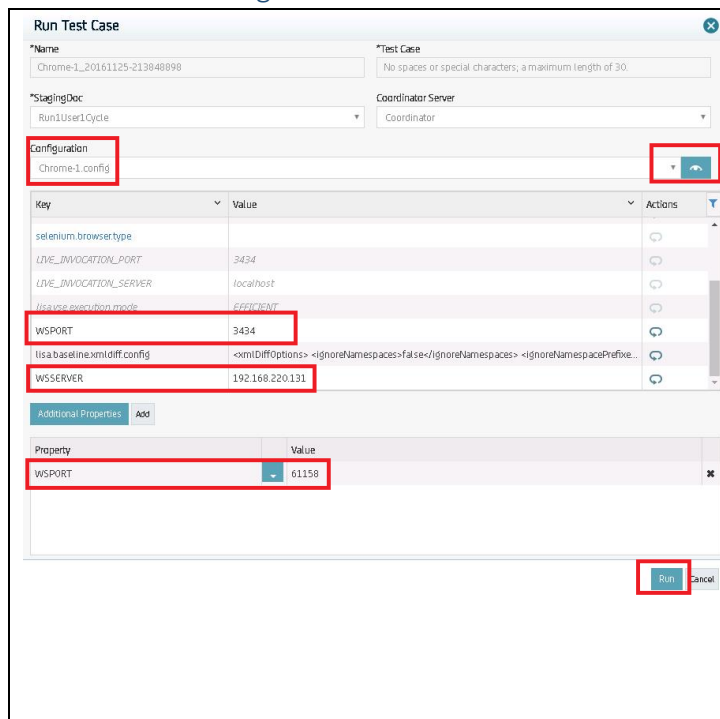


59. From context menu launch 'Run with Options'



60. Review Staging Doc and Coordinator Server settings
61. Select project configuration file of generated test case

## Override Test Configuration

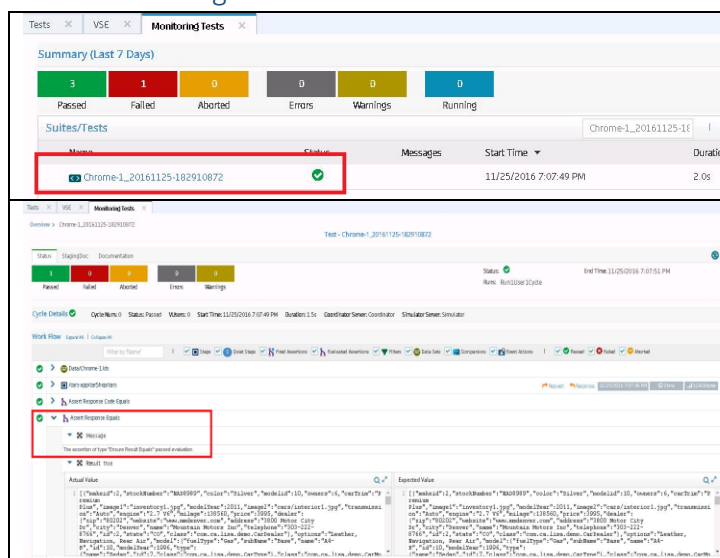


62. Click on 'Configuration Details' icon (at the right end of the configuration file list box)
63. Scroll down to review key 'WSPORT', which should refer to the port of the real service on the local system
64. Click on 'Additional Properties'
65. Click on 'Add' button (close to 'Additional Properties')
66. Under 'Property' change 'key' to 'WSPORT' and under 'Value' change 'Value' to the port where the VS is listening.

We do not need to change key 'WSSERVER', because the VSE, hosting the virtual service, runs on the same system as the real service.

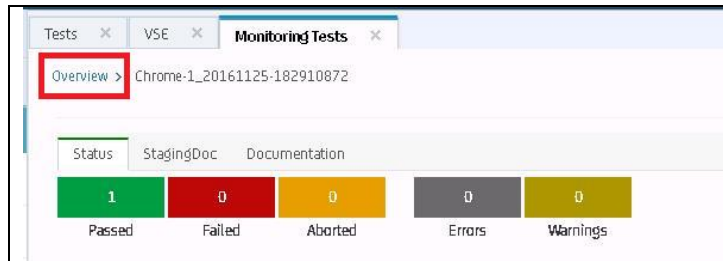
67. Click 'Run'

## Review Test Progress



68. Navigate to 'Monitor > Tests'
69. Review the Status

70. Once completed, drill down into test results by clicking on the test case link



71. Return to 'Monitor > Tests' overview by clicking on the 'Overview' link in the top left corner of the pane