Service Virtualization

Knowledge Transfer Document

Version No. 1.2

REVISION HISTORY

	NAME	SIGNATURE	DATE
Prepared By	Sathya L S Meenakshi N Anjali Rai	Sathya Meenakshi Anjali	6-6-2017
Reviewed By			
Approved By			

VERSION HISTORY

Version No.	Date	Changed By	Changes Made
1	6th Jun 2017	Sathya	Initial Draft Version
2	7th Jul 2017	Sathya, Anjali, Meenakshi	Updated JSON scenarios, JMS, Why SV, benefits of SV

Knowledge Transfer Document

Table of Contents

1. Introduction

Objective

This document will provide the basic understanding of concepts of Service Virtualization & its overview, DevTest tool details, creation of Virtual Services using different methods, Properties & configurations, MQ & JMS queues/topics, REST API, Filters, Assertions, Data Sets, connection of external & internal Database & creation and running of test cases and JSON.

Service Virtualization Overview

Service Virtualization is a practice of capturing and simulating the behaviour, data and performance characteristics of complete composite application environments, so that they react and respond very realistically for development and test teams throughout the software lifecycle, just as the real production environment would.

What is Service Virtualization

- Virtual service is used to mimic/simulate the behaviour of the real system till the real system is available
- Virtual service is used to substitute a real system
- Virtual Service behaves just like how a real system would behave & perform like how the real system would perform
- Service Virtualization acts as a catalyst for Continuous Testing by simulating constrained or unavailable systems

2. Why Service Virtualization

The growing complexity of application architectures—along with globally distributed organizations—means development and testing teams face many bottlenecks and constraints on the road to delivery. These include lack of access to a mainframe partition or ERP systems, unavailable test data and expensive third-party systems. And even more constraints occur when development teams work in parallel and need access to the same environments.

Traditionally, testing teams have had to wait for nearly completed applications to be deployed before proper functional, integration, and performance testing could begin. Distinct project teams might produce different components of a system or application, one piece at a time, and then assemble them into a single working product before allowing the testers to have their way with it. It's logical, it's linear, and it's slow. Furthermore, current processes and tools are failing to overcome these limitations such as:

1. **Unavailable systems:** Systems become constrained due to schedules, security restrictions, contention between teams or because they are still under development.
2. **Poor performing applications:** Downstream systems and mockups may not provide the functional behavior or performance response needed, network connections in the test lab do not reflect production network conditions and end-user performance suffers.
3. **Costly third-party access fees:** Developing or testing against cloud-based or other shared services can result in costly usage fees.

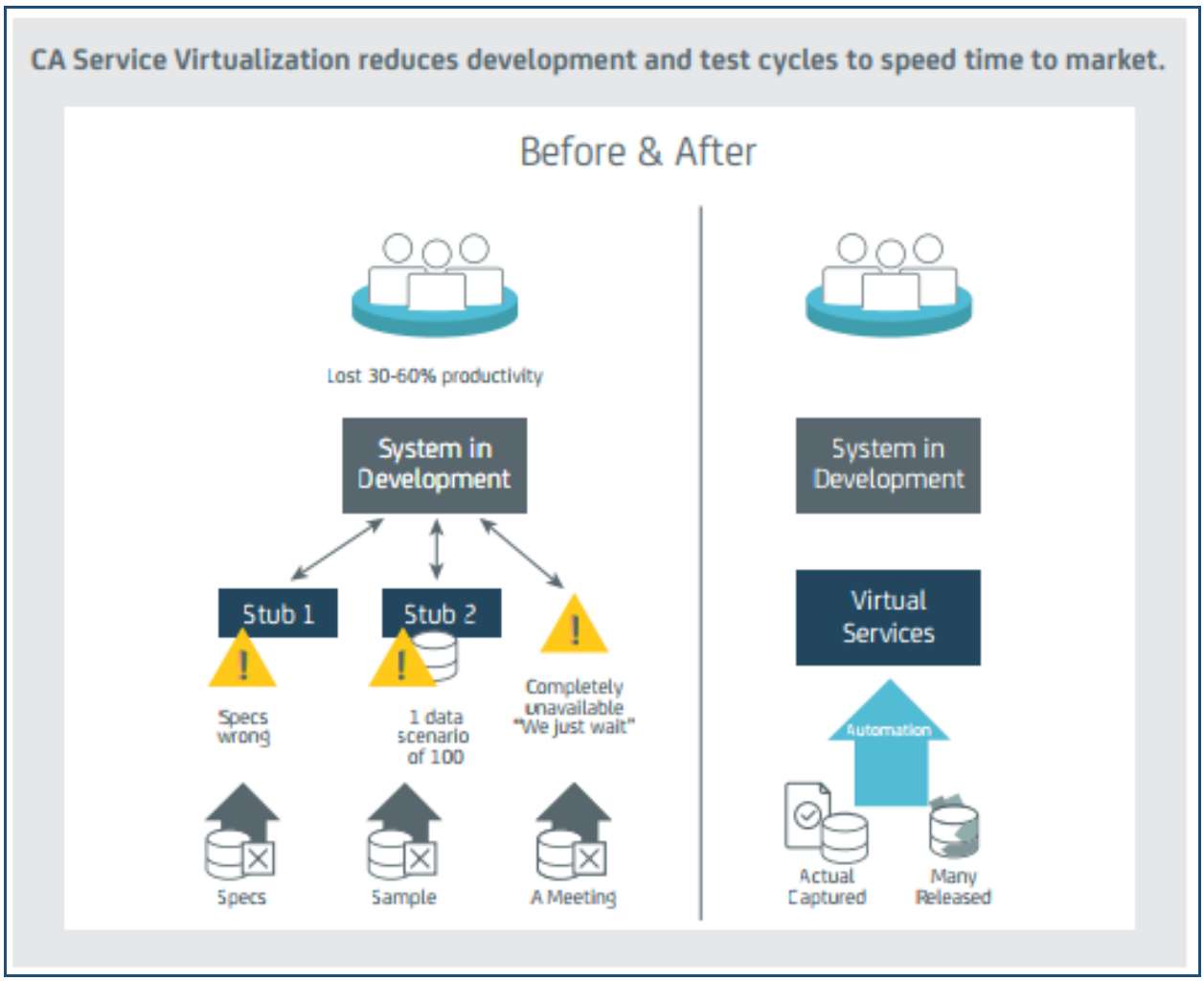
CA Service Virtualization eliminates these constraints by creating simulations of essential systems and making them available throughout the SDLC. Developers, testers and performance teams work in parallel, leading to faster delivery, lower costs and higher quality of innovative new software applications. It helps overcome:

1. **Infrastructure constraints**: Reduce the amount of hardware and software needed for a highly scalable, productive, unconstrained development and testing environment.
2. **External service constraints:** Eliminate barriers and streamline development by simulating dependent systems including mainframes, external service providers and ERP systems.
3. **Parallel development constraints:** Allow projects to be developed in parallel instead of a classic serial, waterfall model, accelerating development and time-to-market.

4. **Test scenario constraints**: Dramatically simplify the creation and management of development and testing processes, such as test data, system configuration, and other non-value-add activities.

3. Benefits of Service Virtualization

1. **Parallel development and testing:** Enable multiple development and testing teams to work in parallel, eliminating schedule bottlenecks and speeding time to market
2. **Infrastructure requirement reduction:** Eliminate much of the concurrent demand for environments created by high-velocity development and test processes
3. **"Shift Left" and test more:** Test earlier in the software lifecycle when issues are easier and less expensive to resolve
4. **Performance readiness:** Load test at the component level with production-level conditions
5. **Elimination of costs for third-party services:** Avoid costs by simulating needed third-party services



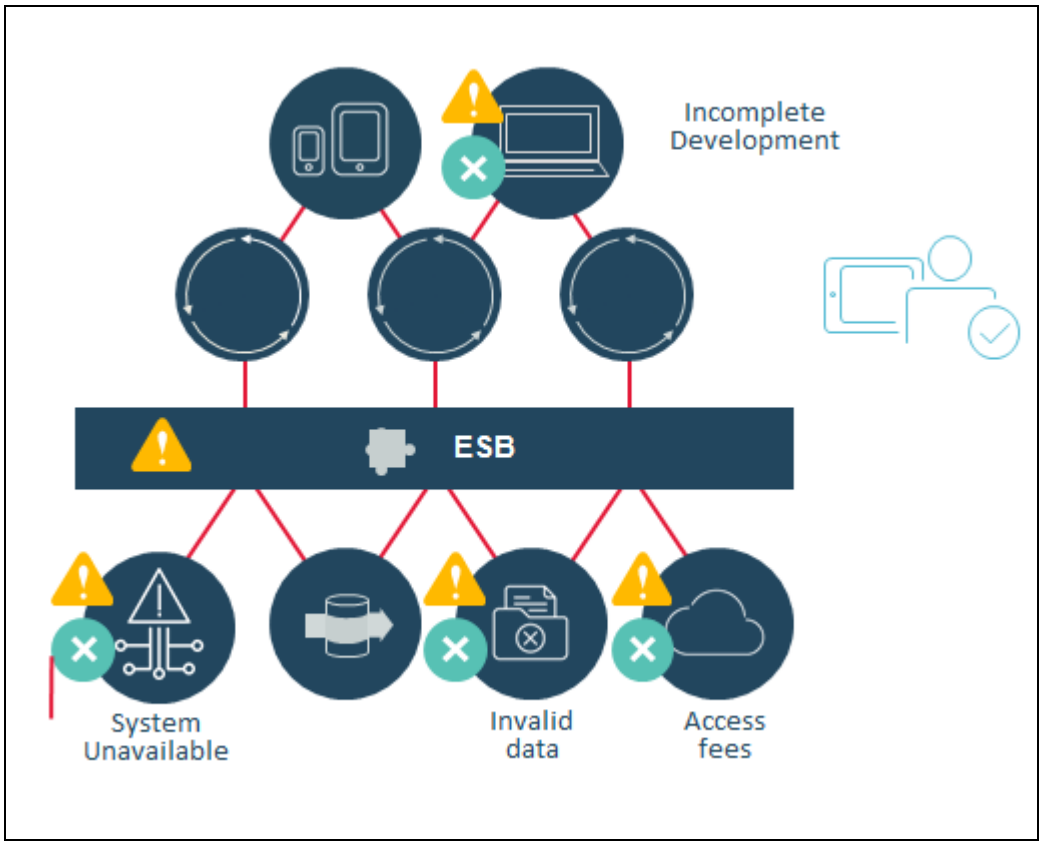
References:

https://www.ca.com/content/dam/ca/us/files/white-paper/the-why-where-and-how-of-service-virtualization-adoption.pdf

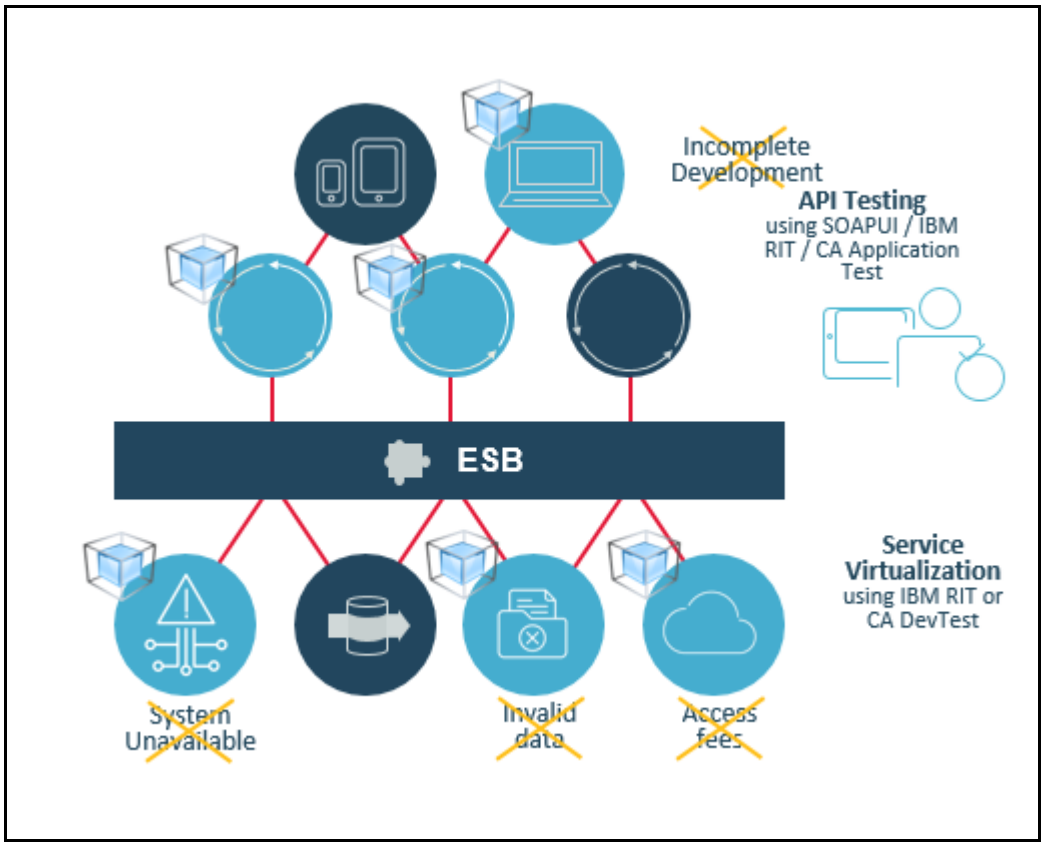
https://www.ca.com/content/dam/ca/us/files/data-sheet/ca-service-virtualization.PDF

4. Use of Service Virtualization

Why it is needed:



Eliminate Dev/Test Constraints:



5. Service Virtualization Use Cases

- **Faster time-to-Market**

  By eliminating constraints on every phase of the SDLC, Service Virtualization can accelerate delivery time for software projects by 25-50%.

- **Shift Left**

  Move software development into parallel and test and validate sooner in the software lifecycle where it is less expensive and issues are easier, less disruptive, and less expensive to resolve.

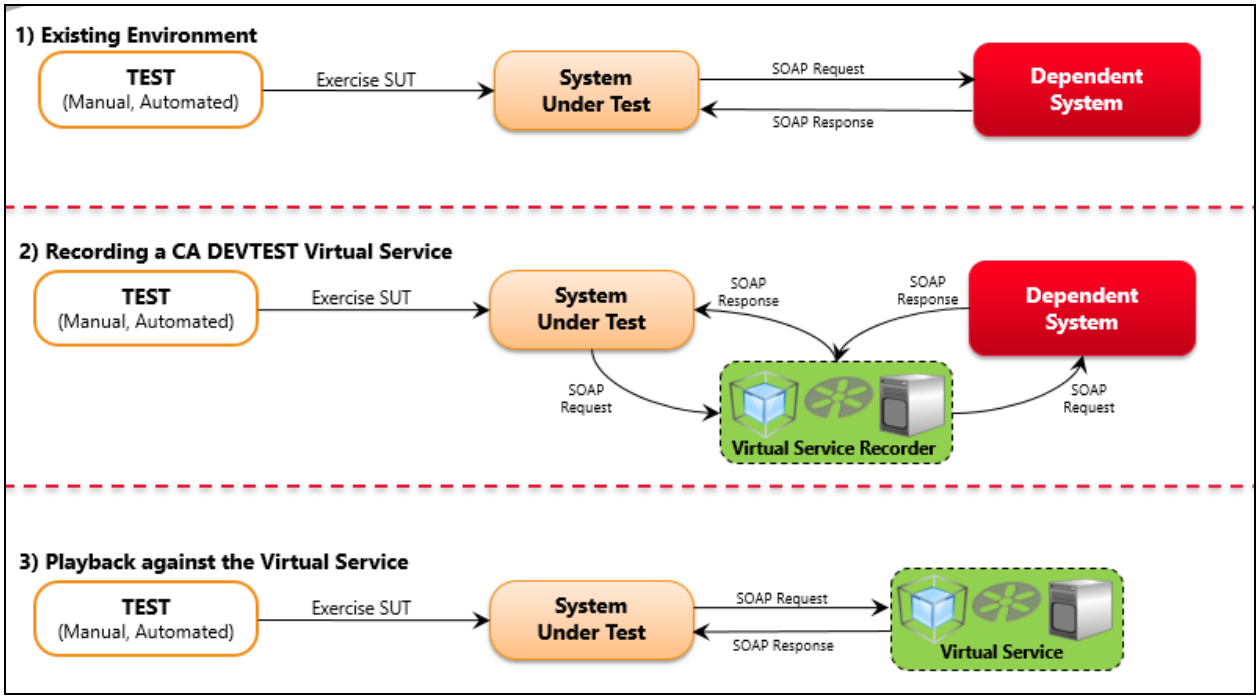
- **Reducing demand for lab infrastructure and software**

  Eliminate much of the concurrent demand for environments and hardware requirements agile methodologies create.

- **Accelerated Quality**

  Use SV to deliver a better end –user experienced by improving application performance and quality.

6. How SV Works



7. CA SV Server Components

- **Enterprise Dashboard:** Monitor enterprise activity. Generate the Usage Audit Report
- **Registry**: A central location for the registration of all CA SV Server and CA SV Workstation components
- **VSE**: Used to deploy and run virtual service models.
- **Simulator Server**: The simulator runs the tests under the supervision of the coordinator server.
- **Coordinator Server**: The coordinator receives the test run information as MAR files, and coordinates the tests that are run on one or more simulator servers.
- **DevTest Portal**: DevTest Portal is a web-based application that provides simpler access to the most commonly used workflows for DevTest products

8. CA Dev Test Client

An integrated development environment(IDE) where test case assets and virtual service models are created and edited. You can run test cases and models locally in the workstation or you can stage them for a remote execution. CA SV Workstation must be installed on desktop computers for users who author CA SV test and virtual model assets. Any number of workstations can attach to the registry and can share the server environment.

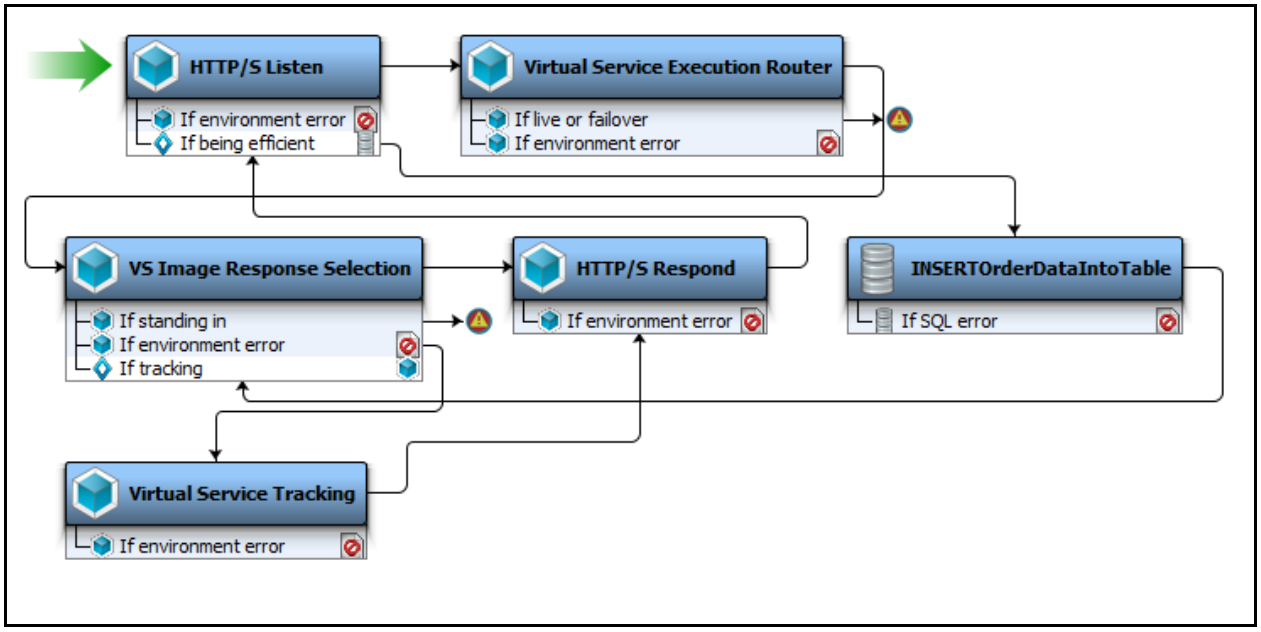
DevTest Coupled with below products:

- CA Application Test
- CA Service Virtualization

9. Virtual Service Model (VSM)

- A Virtual Service Model (VSM) is a specialized type of test case that becomes the endpoint of a virtualized service
- Virtual Service Image recording creates a Virtual Service Model (VSM) with seven steps or nine steps, depending on the option chosen (More Flexible or More Efficient). Sometimes you would like to edit the VSM by editing generated steps or adding more steps

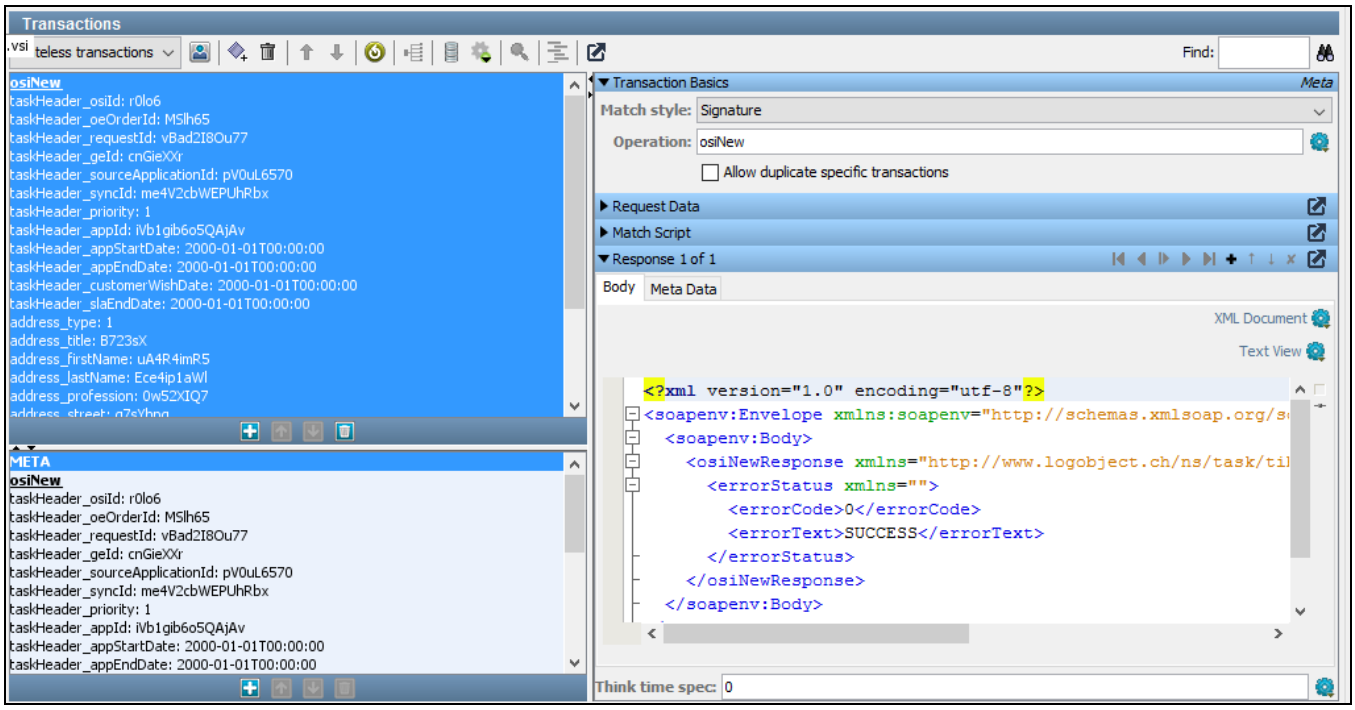
Sample:



10. Virtual Service Image (VSI)

- The Virtual Service Image Recorder generates service images
- Service images pretend to be what recorded informs of transaction
- It is the important component VSM
- VSI keeps the information regarding request-response

Sample:



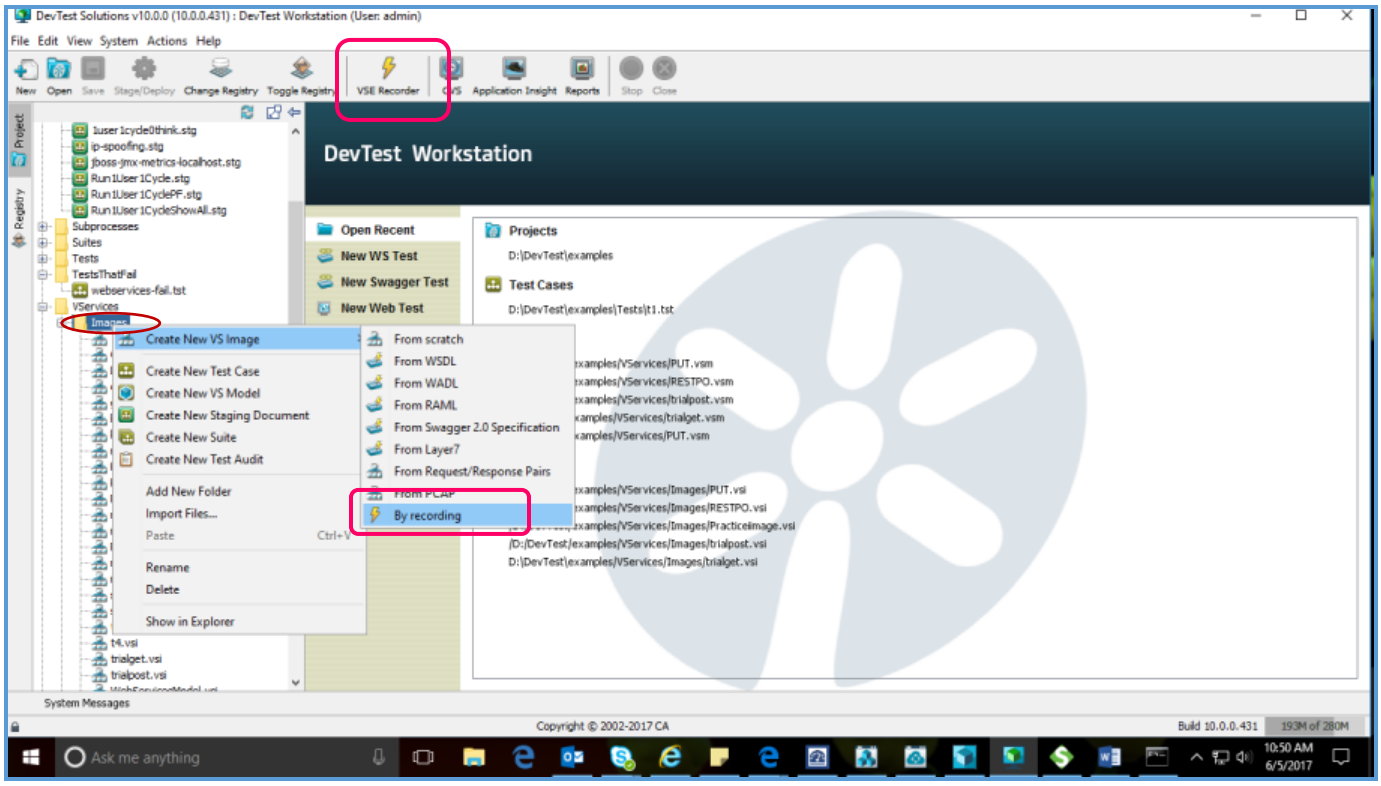
11. Creation of Virtual Service by Recording

The Virtual Service Image Recorder generates service images, which pretend to be what you recorded. DevTest Workstation lets the user to create a virtual service by recording the traffic between a client and a server. This procedure assumes that a virtual service environment (VSE), DevTest Demo Server are running.

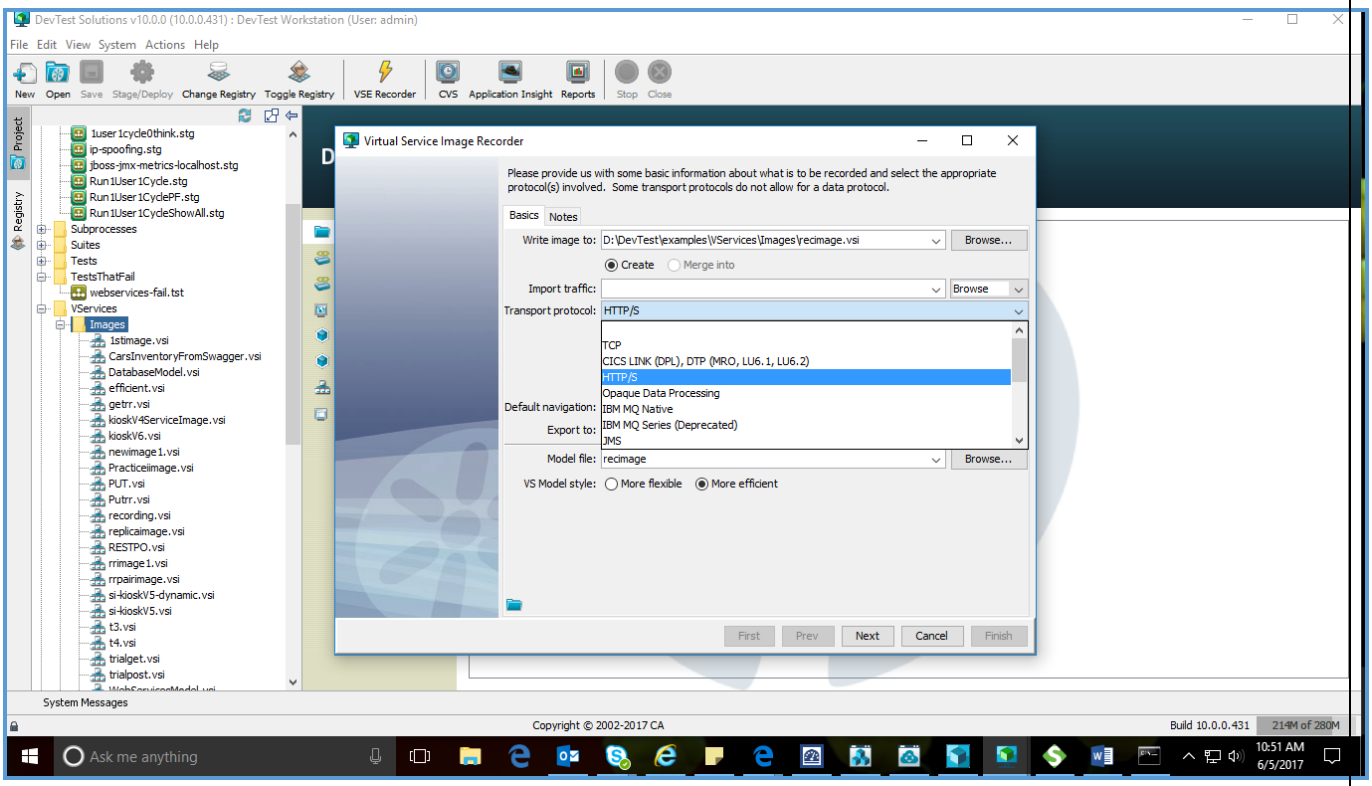
*In this KT document **LISA Bank** application is used as an example.*

The steps are as follows:

1. Click VSE Recorder icon on the main toolbar or Right-click on Images Folder > Create New VS Image > By Recording option

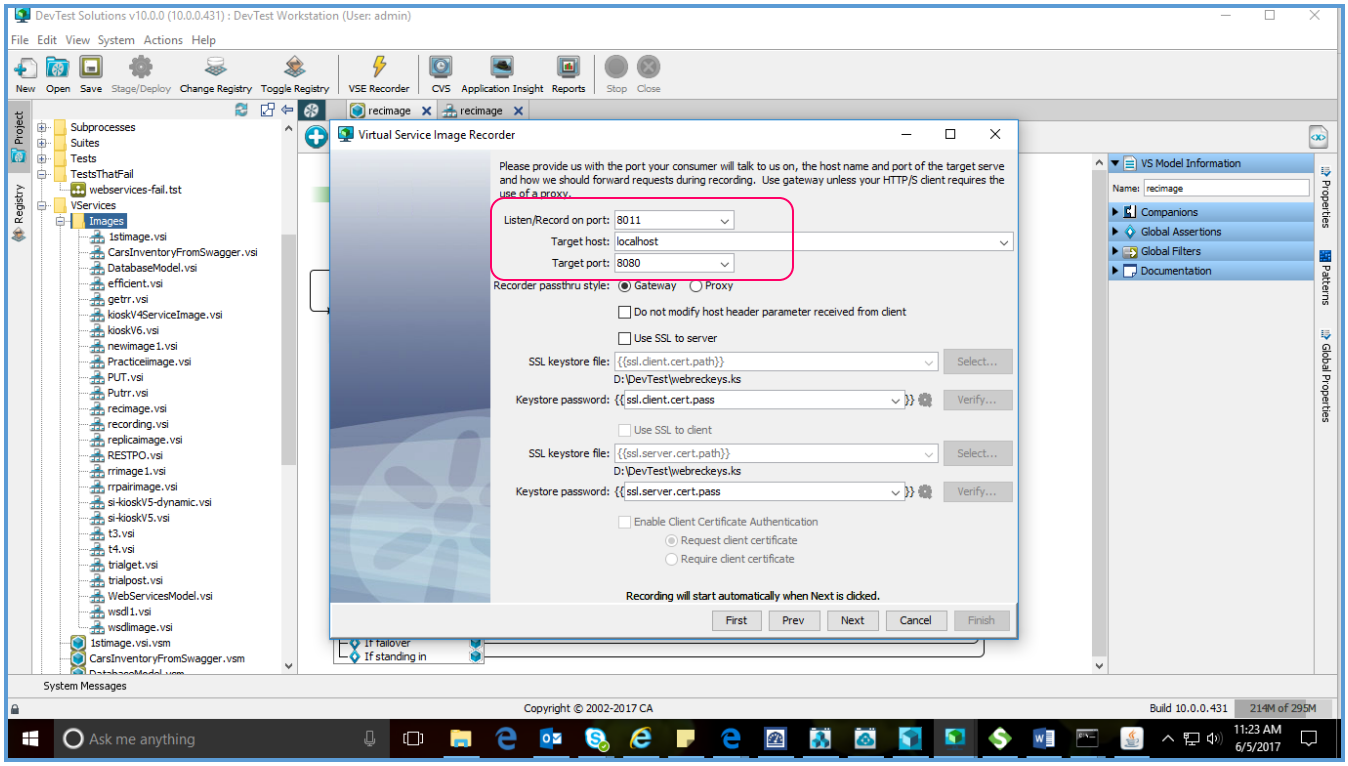


2. The Virtual Service Image Recorder window opens. Provide **Write Image to** Name for the VSI, select appropriate **Transport protocol** from Drop down list & **Model file** name & then click on Next button



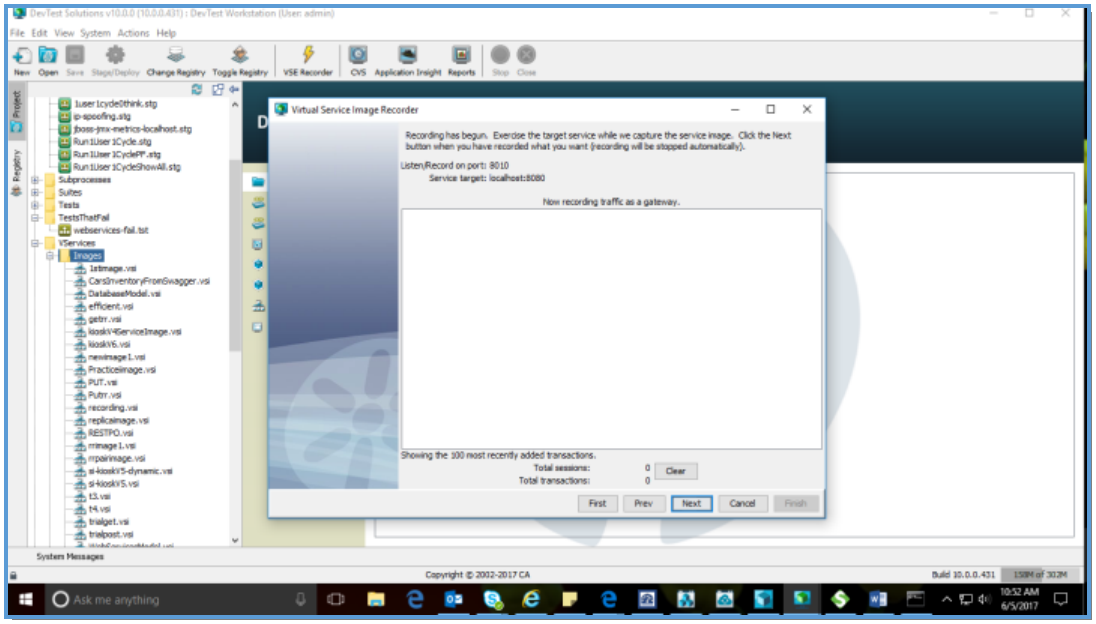
Note: To avoid confusion give same name to the image as well as model

3. Enter values for the **Listen/Record on port, Target host & Target port** & click on Next button

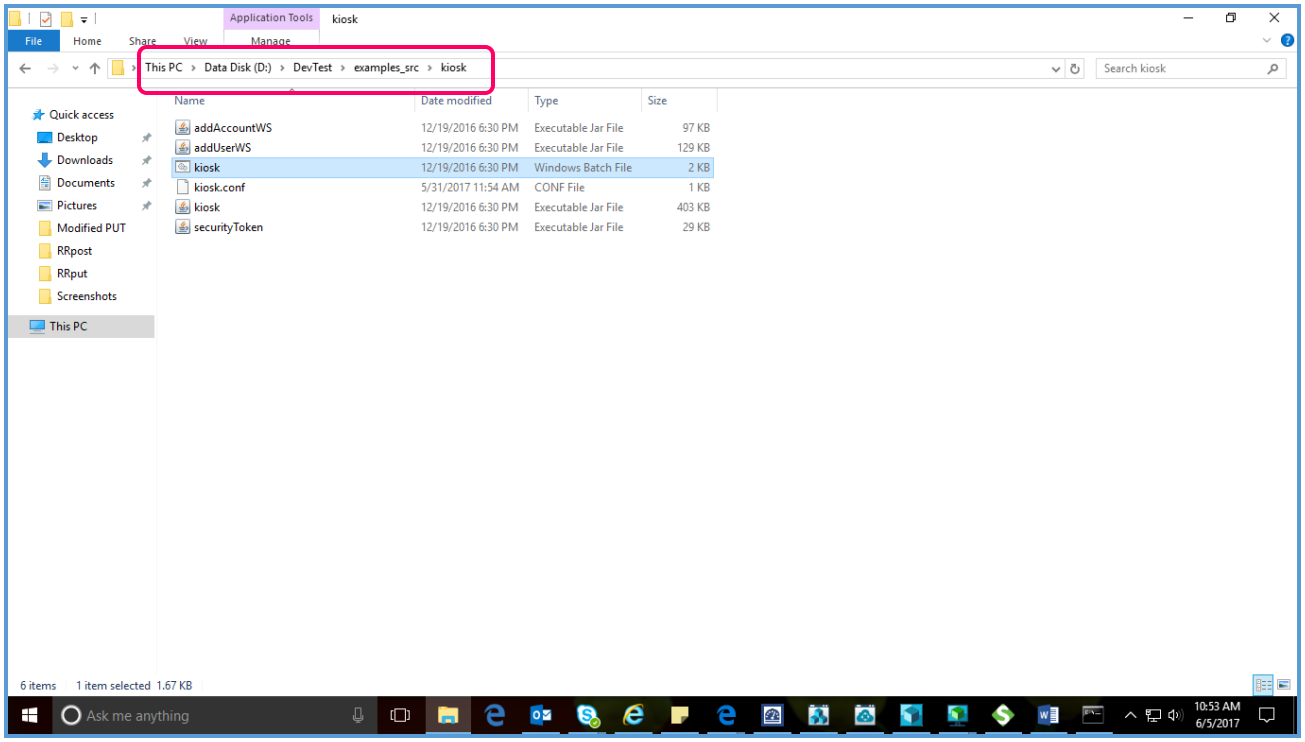


Note: Whenever a new service image is created via recording, the listening port should be different

4. The VS Image Recorder starts recording the traffic



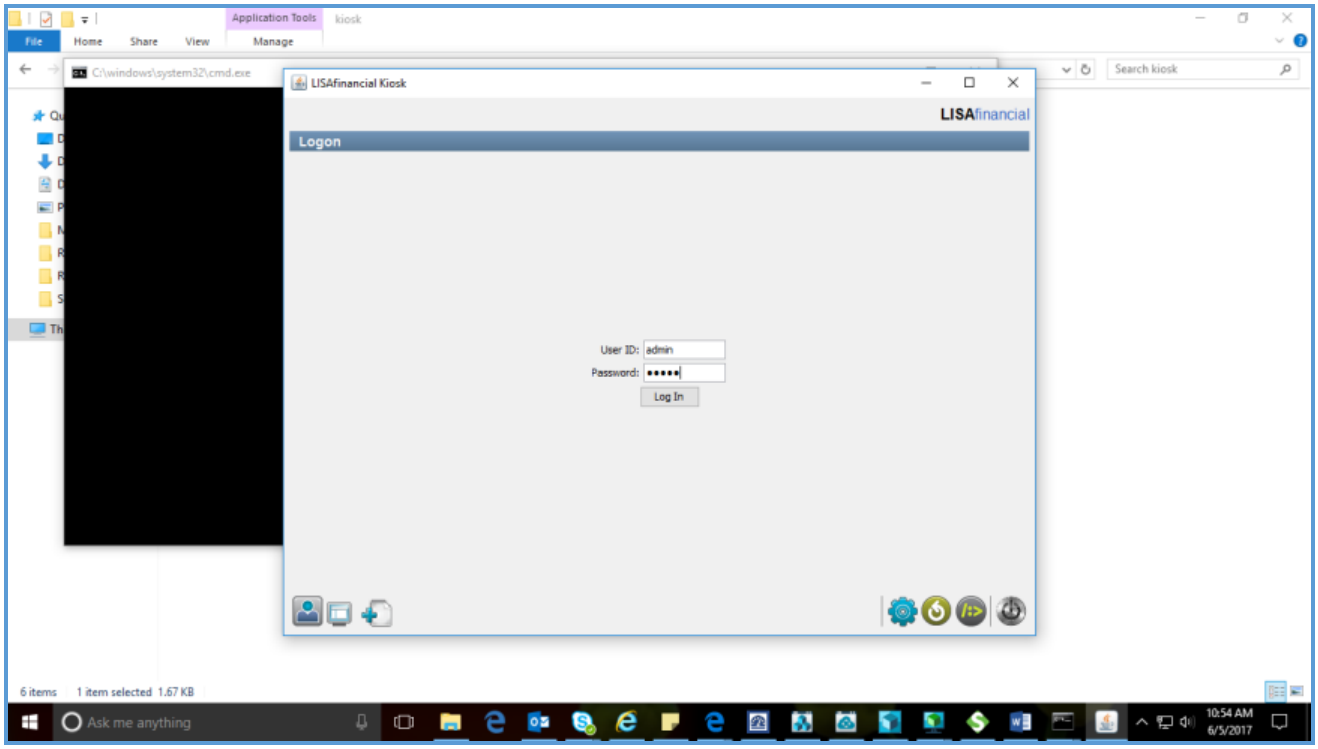
5. Navigate to the location where the **LISA bank application** is present



6. LISA Bank application screen

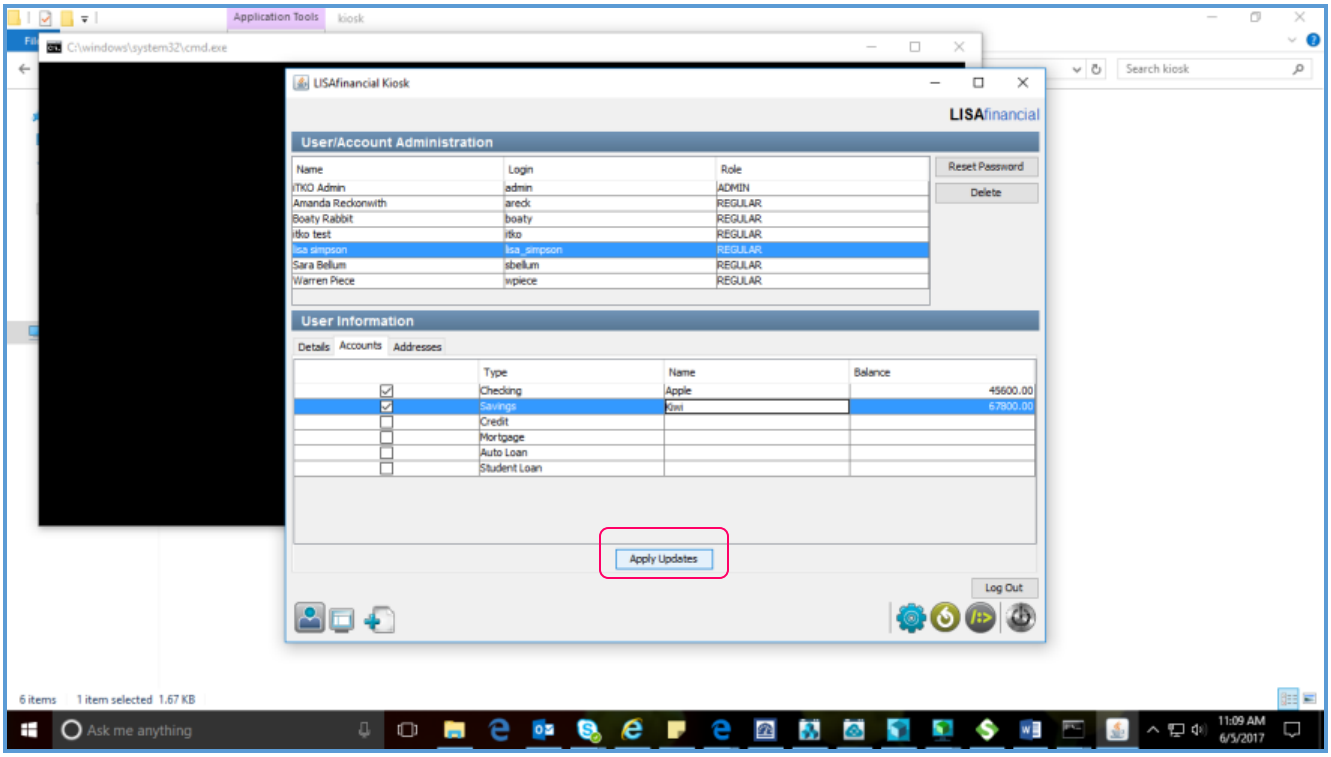
7. Click on settings/retry and enter localhost: **XXXX** (Target port) and login with appropriate username and password to login

Login screen

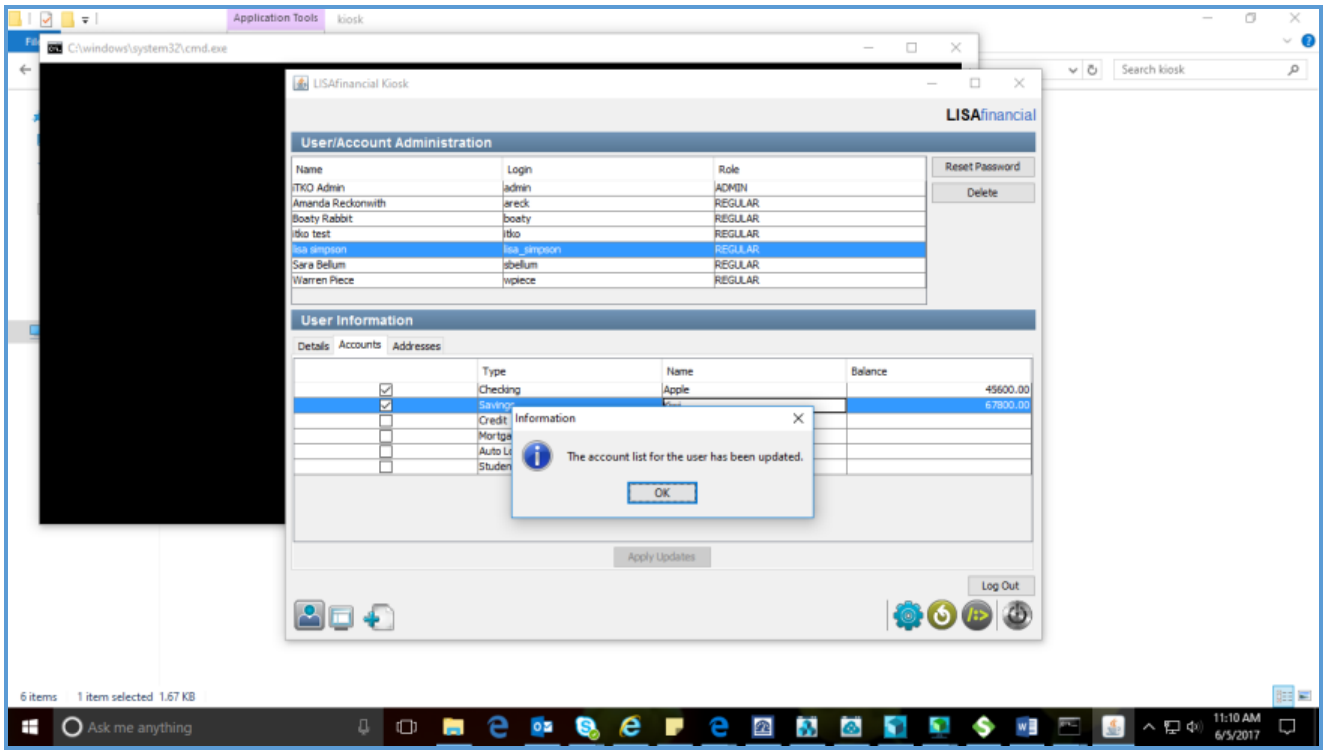


8. Update the following settings for accessing the **LISA Bank app.**

   Select Account Name & click on Accounts tab from User Information section and click on account type check boxes and update Name & Balance column too. Click on Apply Updates button.

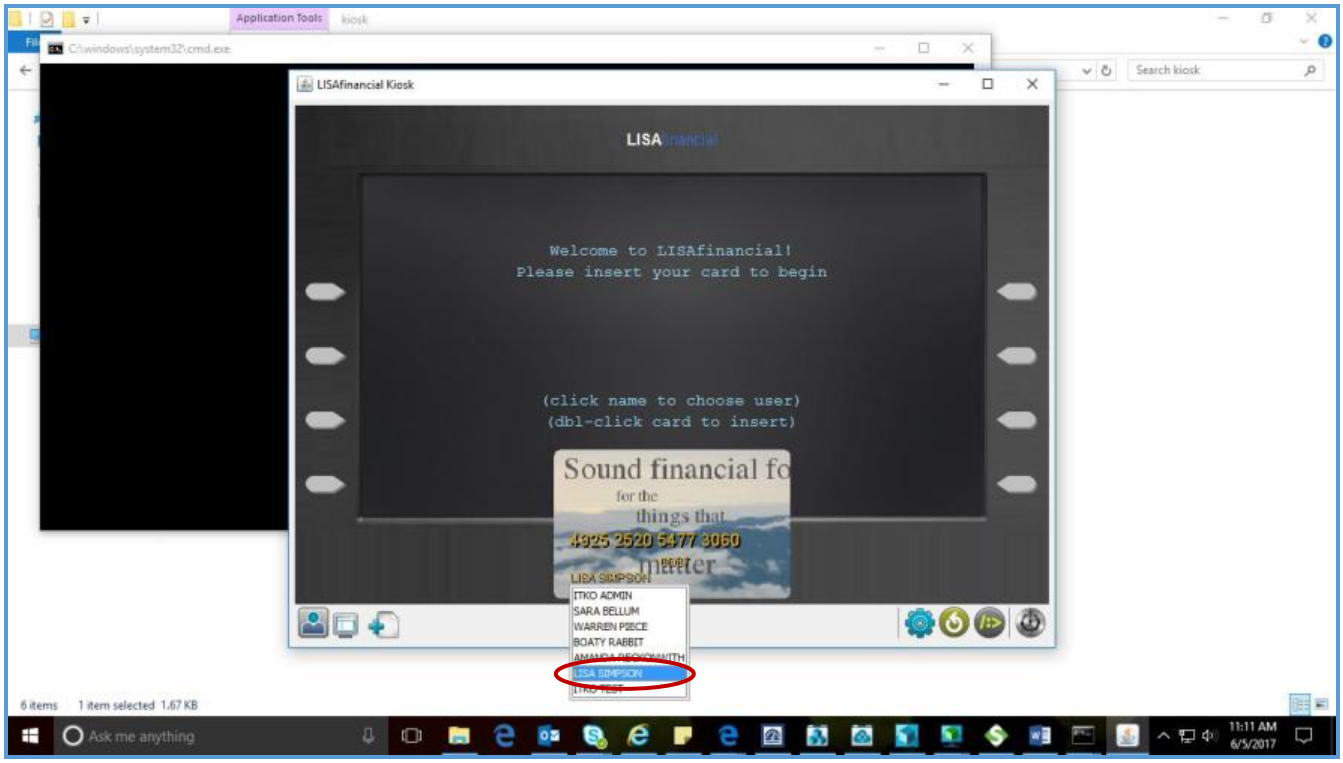


9. Account list for the User Account has been added

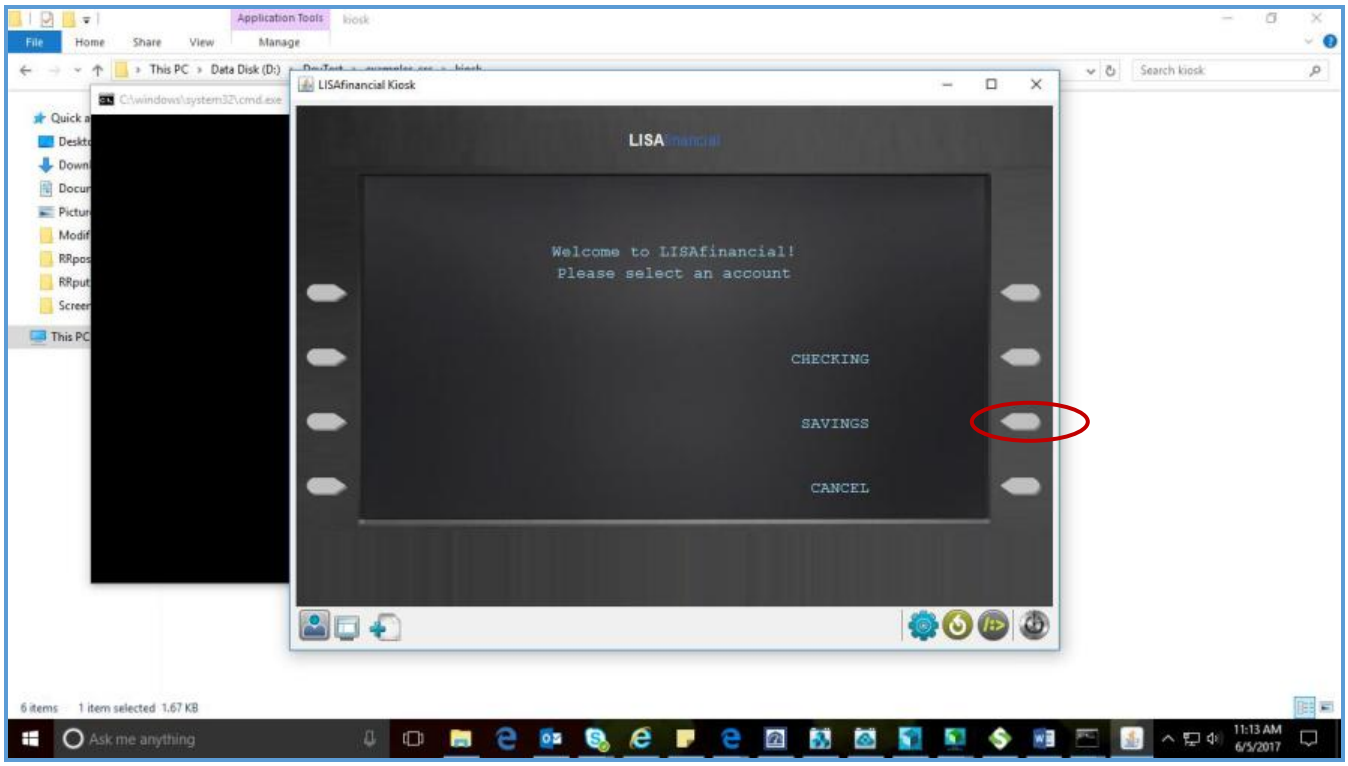


10. In order to begin transaction, click on settings again and change Target Port to the listening port which we recorded in step 3

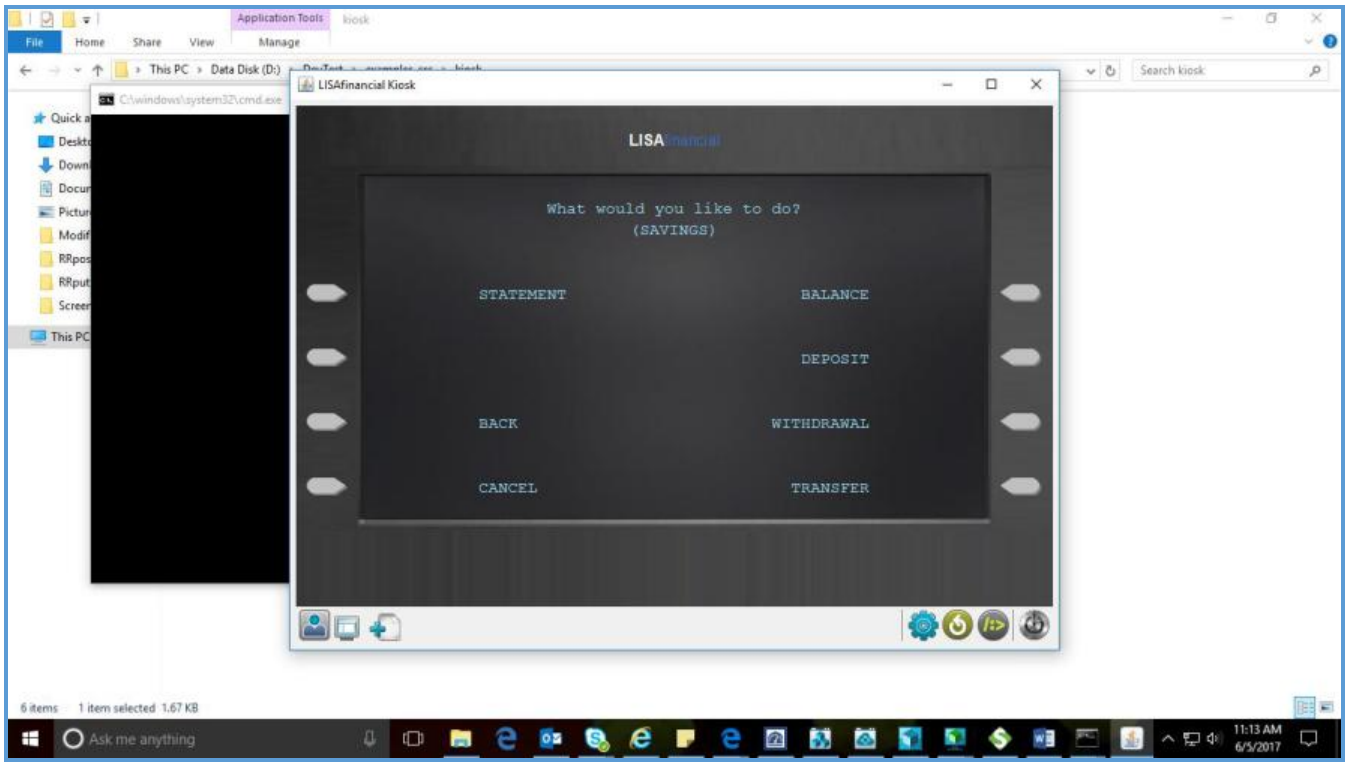
11. Double click on card, Select the User Name & enter password to login



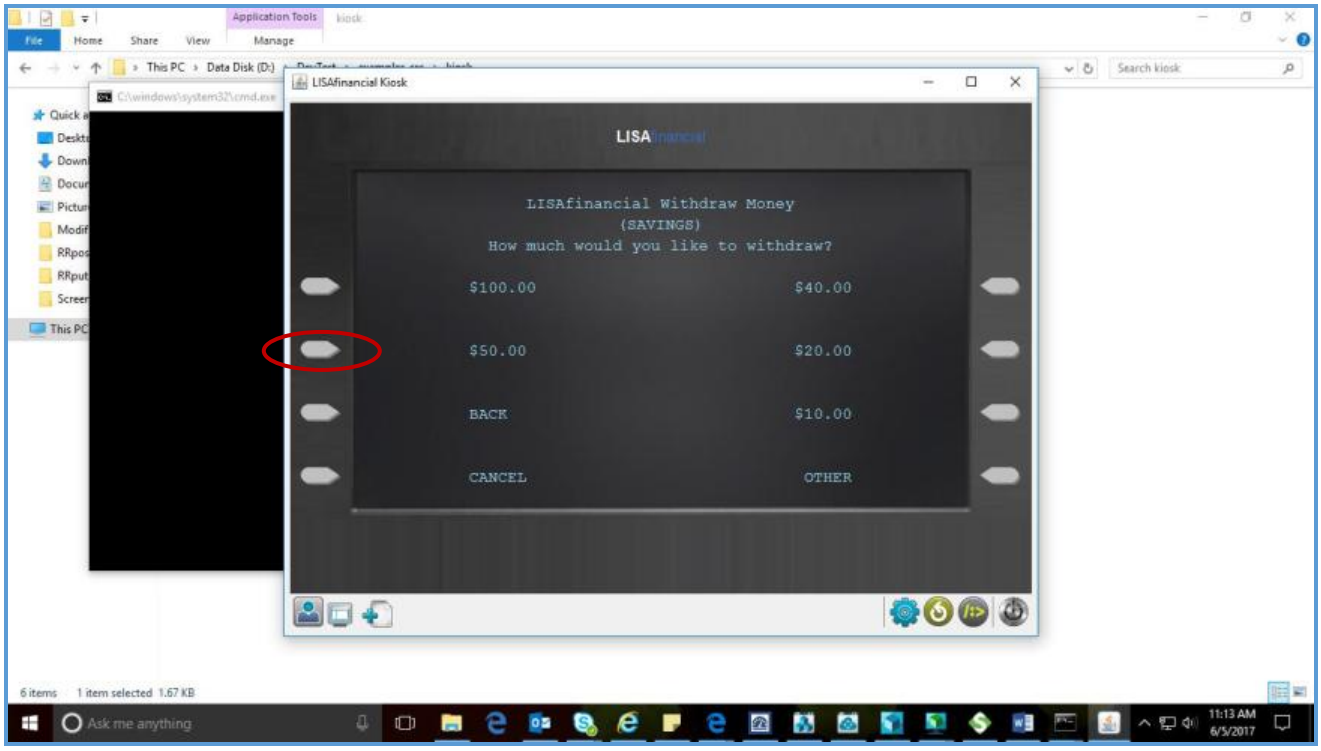
12. Click on the transaction icons to record transactions. In this example saving transaction is performed.



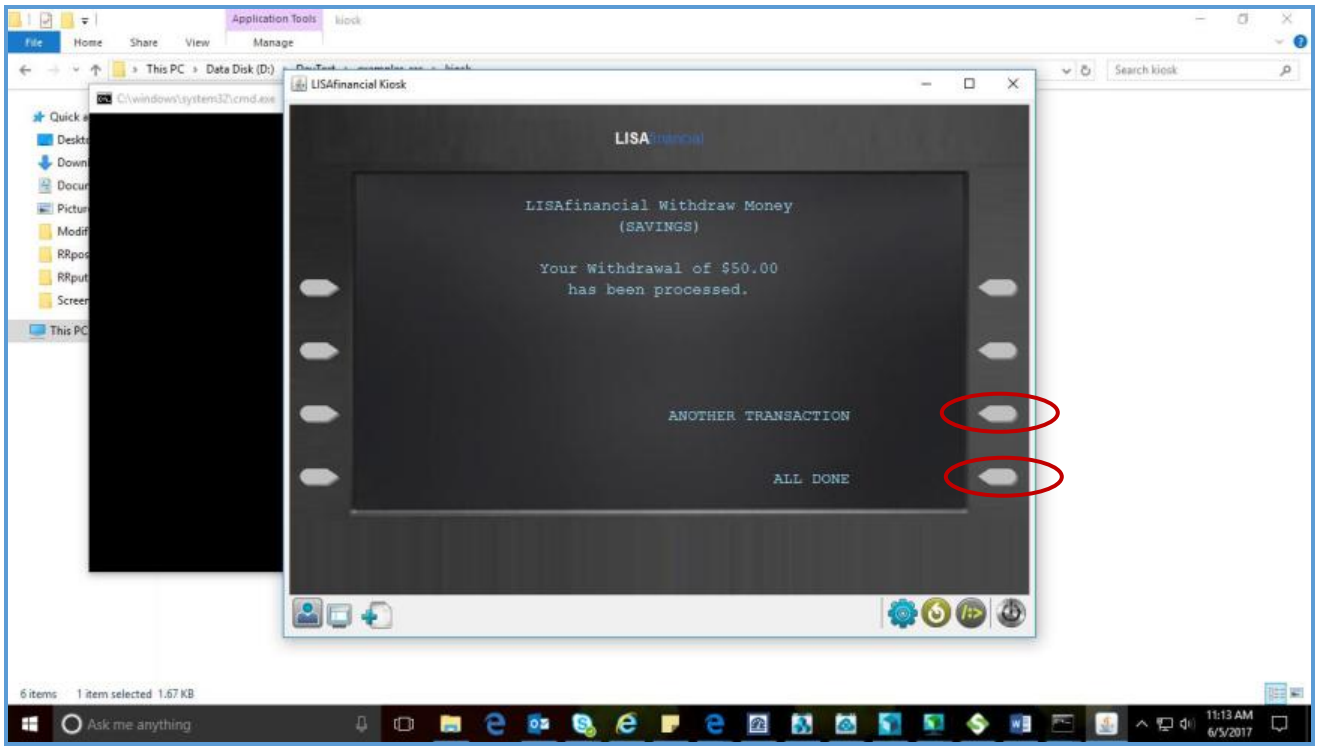
13. Savings screen displayed



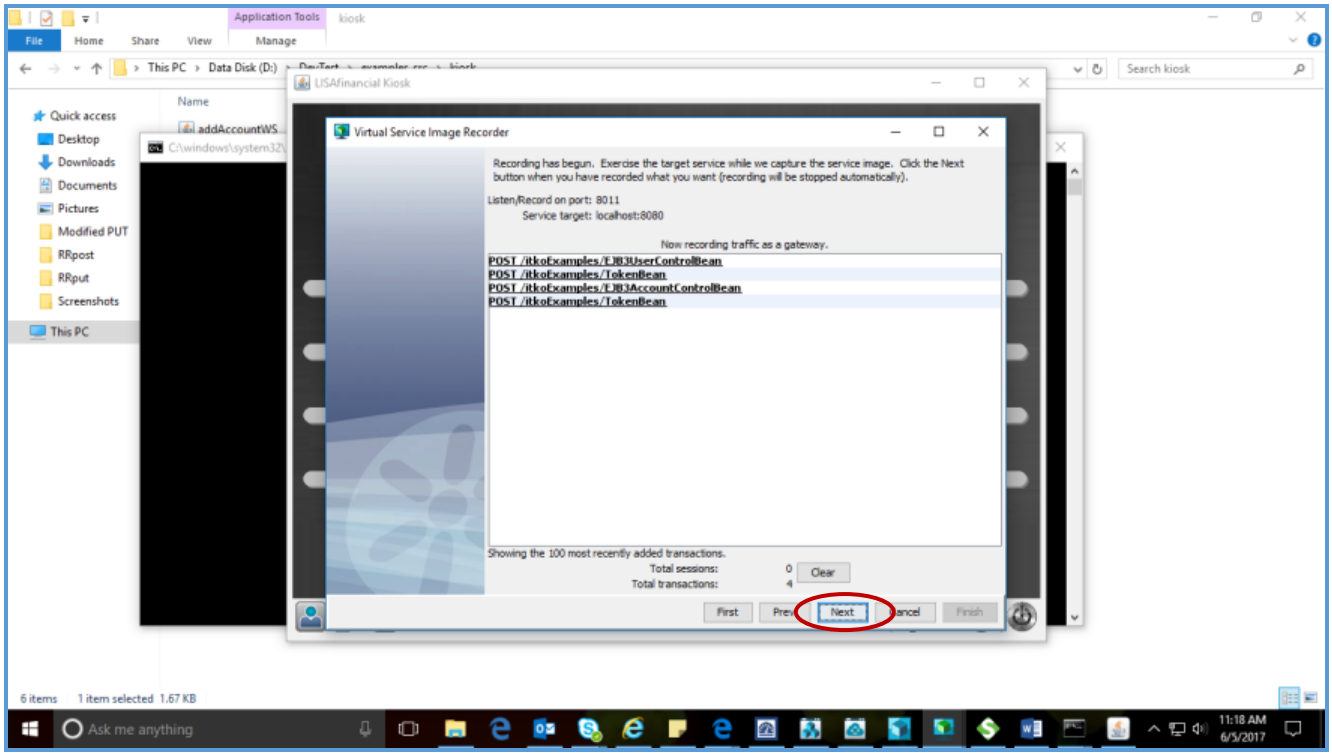
14. Select the operation to be performed from savings.



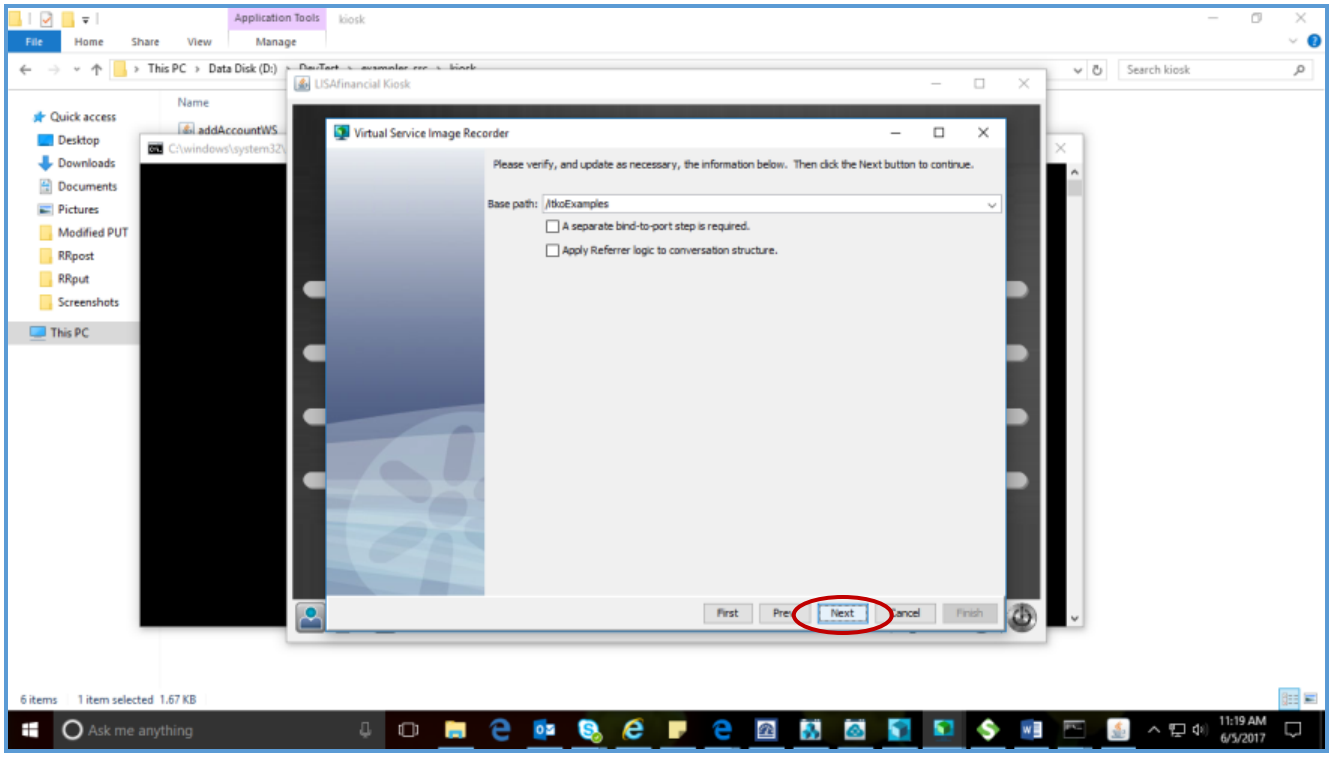
15. Click on Another Transaction record some more transaction in same manner or select All Done option



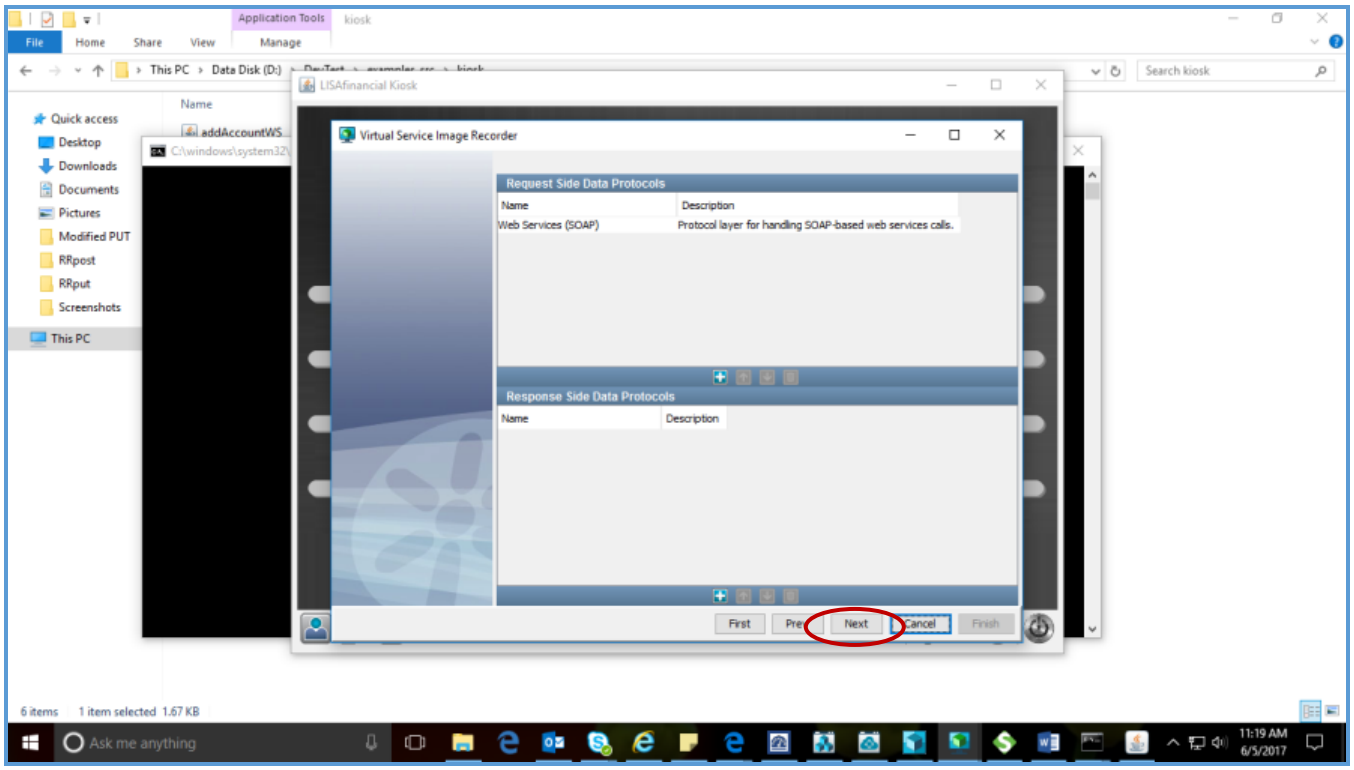
16. Go back to VSI Recorder to view the recorder transaction, click on Next button



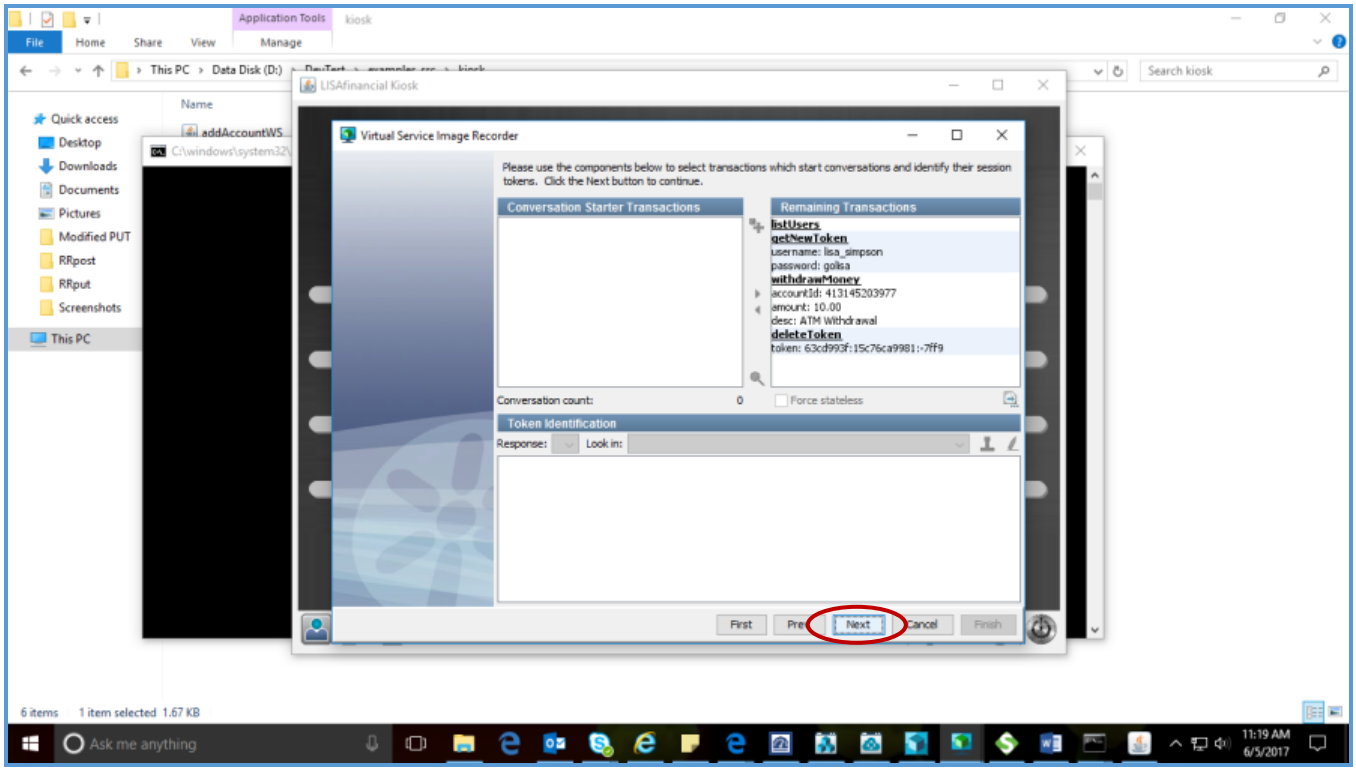
17. Click on Next button (base path displayed)



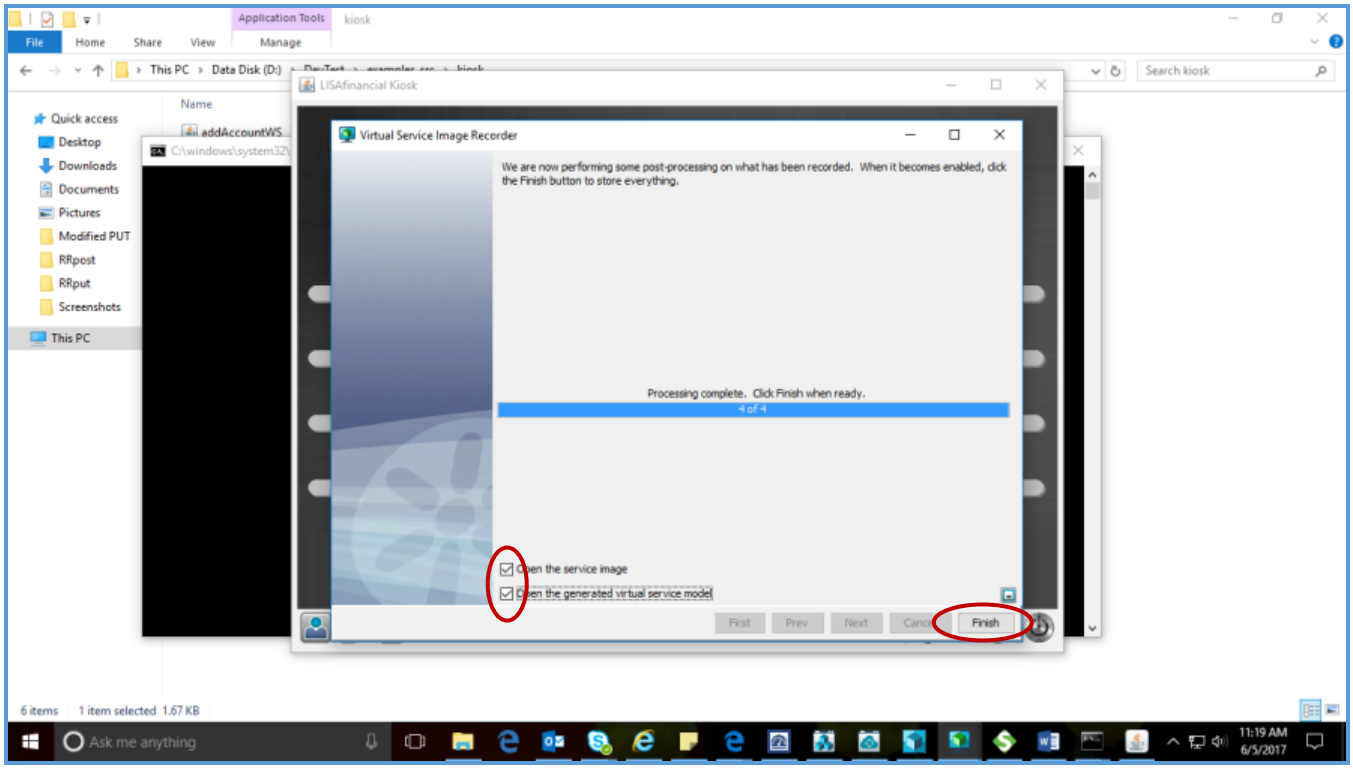
18. Click on Next button (Data protocols are automatically selected)



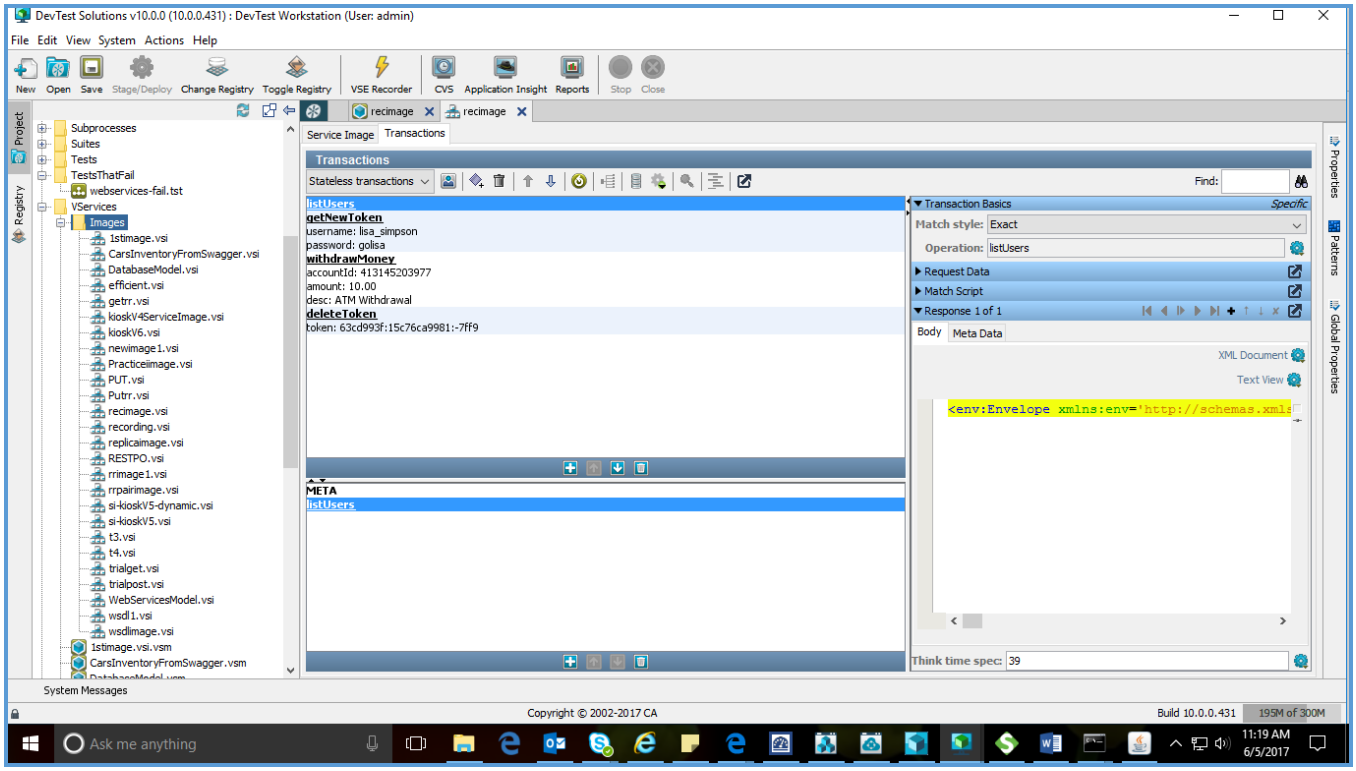
19. Transaction details are dispalyed



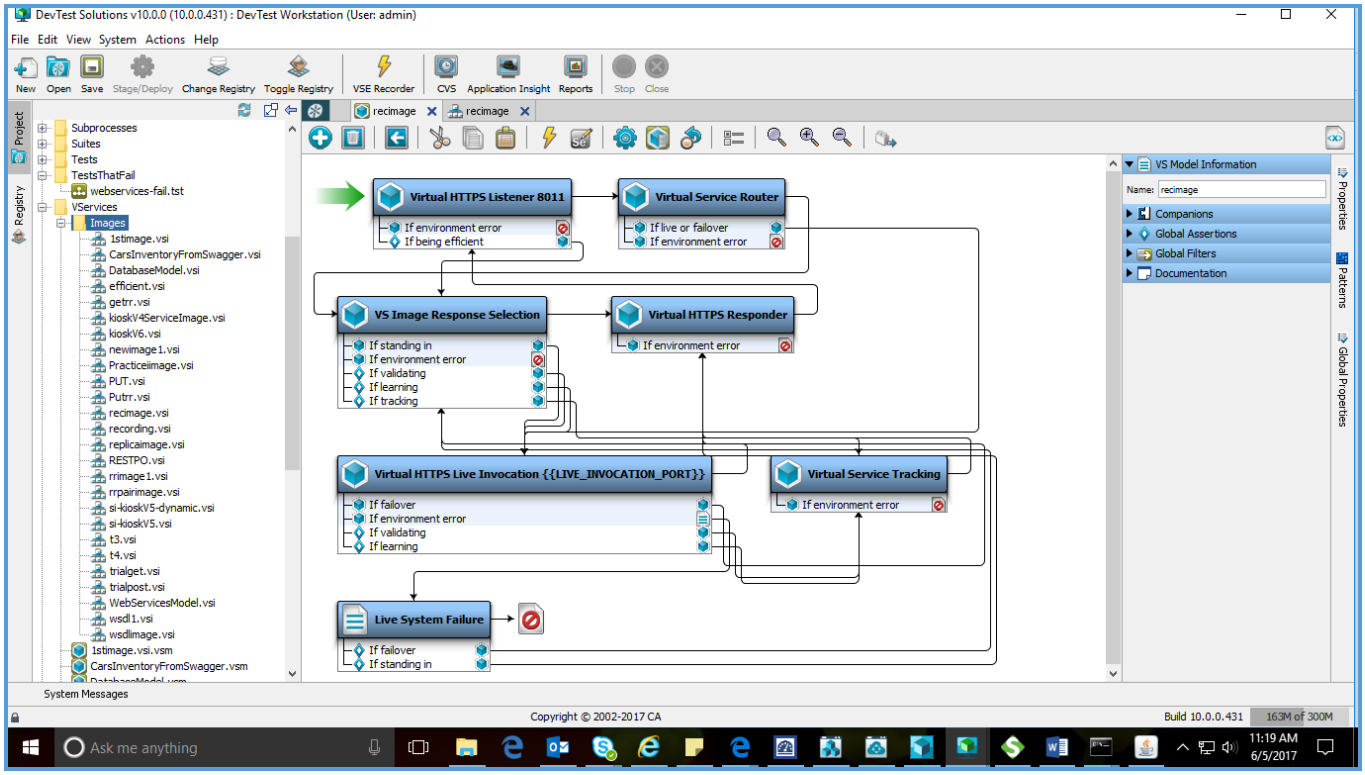
20. Click on checkboxes to open VSI & VSM and click on Finish button



21. Recorded VSI



22. Recorded VSM



23. You can execute the VSM using SoapUI by copying the Request and executing it with correct port and base path that are displayed in VSM by running ITR

12. Creation of Virtual Service using RR Pairs

1. A request/response pair consists of one request file and one or more response files
2. When you import a request/response pair, the request is processed against the virtual service
3. The request/response pair must be in text or XML format
4. Example of file names should be:
    a. depositMoney-req.xml
    b. depositMoney-rsp1.xml
    c. depositMoney-rsp2.xml
5. The request file name must include a prefix followed by the string –req
6. The response file name must include the same prefix followed by the string -rsp. As the preceding example shows, you can provide multiple response files by adding a number after the string -rsp
7. Sample:

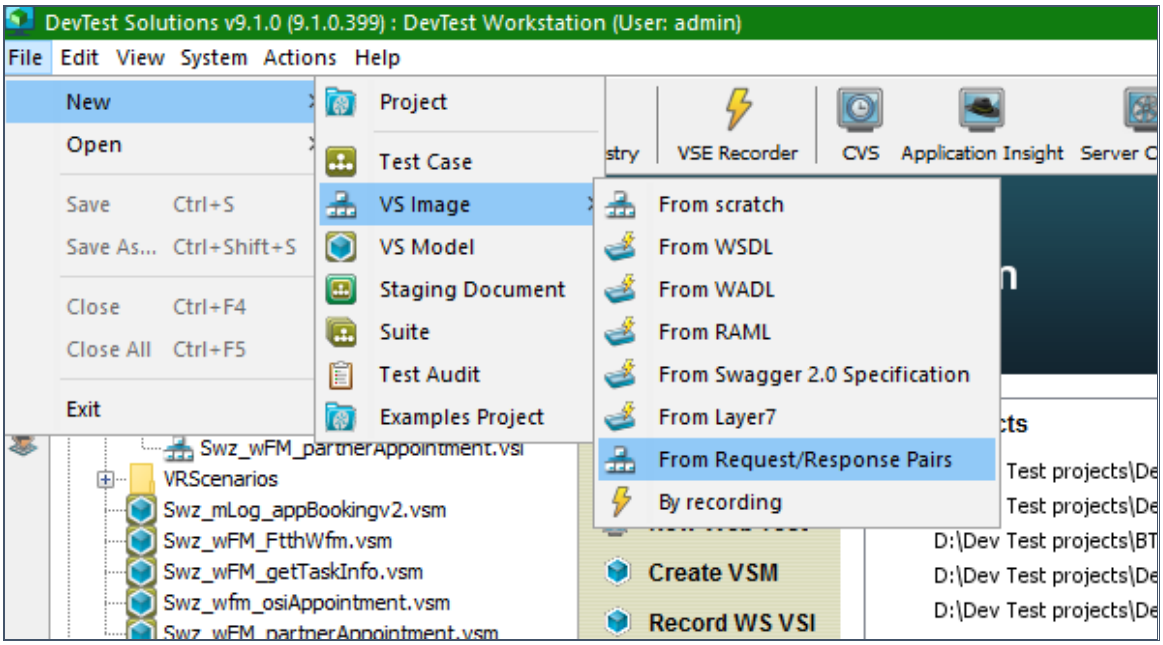
Sample Files are embedded here:


bookappointment-req.xml

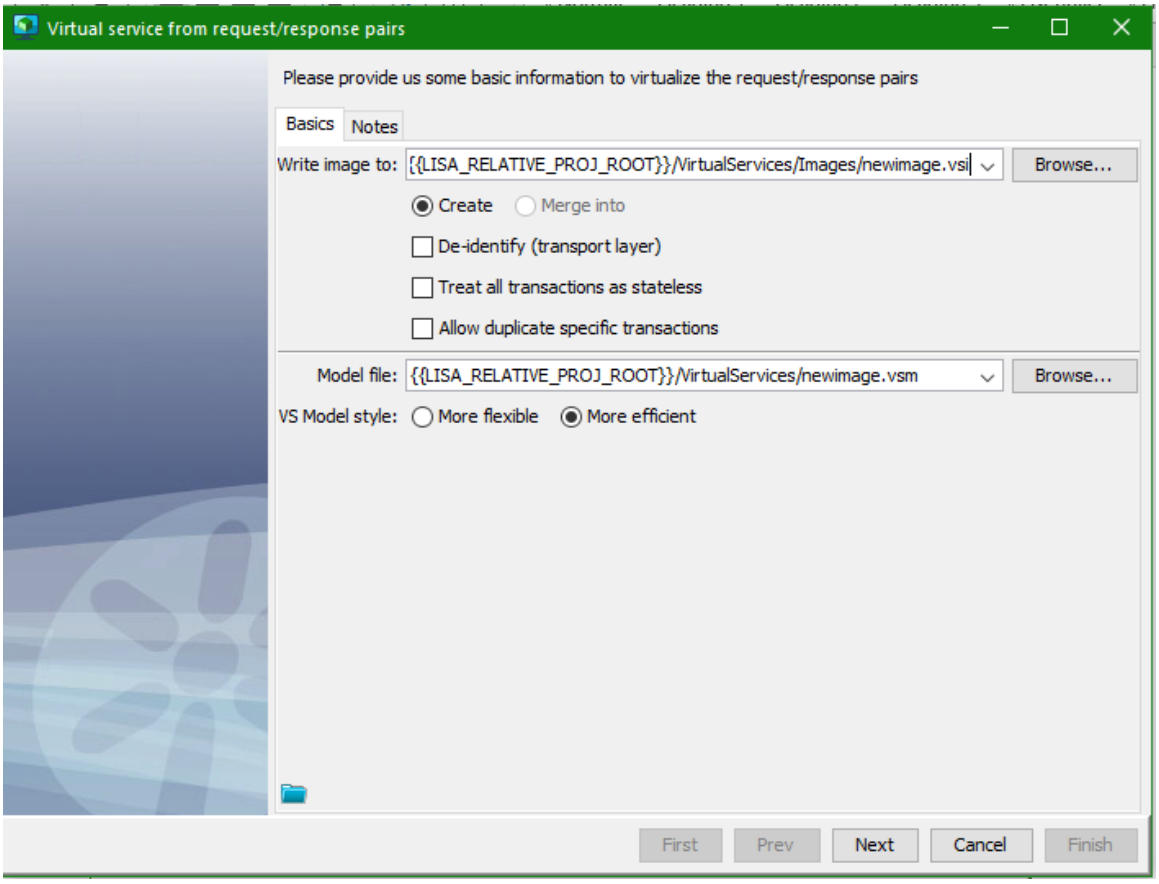

bookappointment-rsp.xml

The steps are as follows:

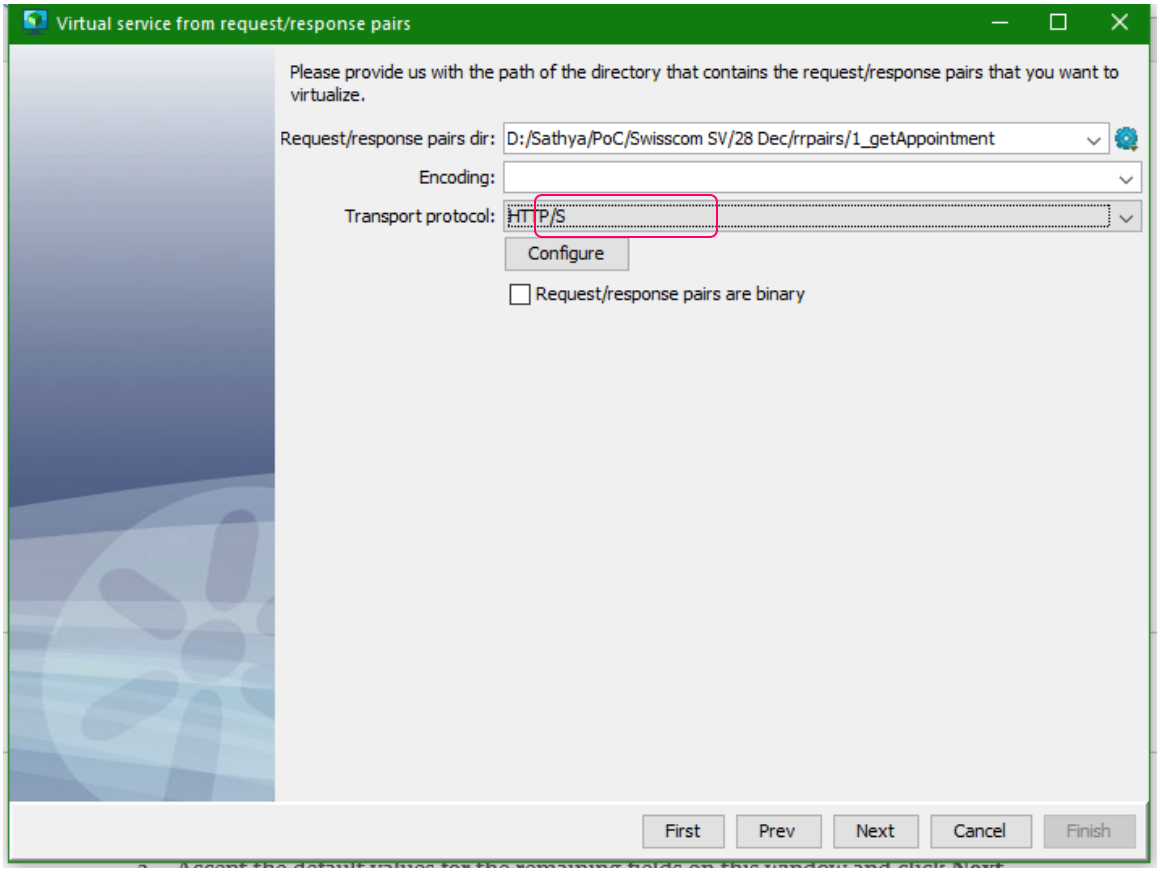
8. Select File > New >VS Image > From Request/Response Pairs



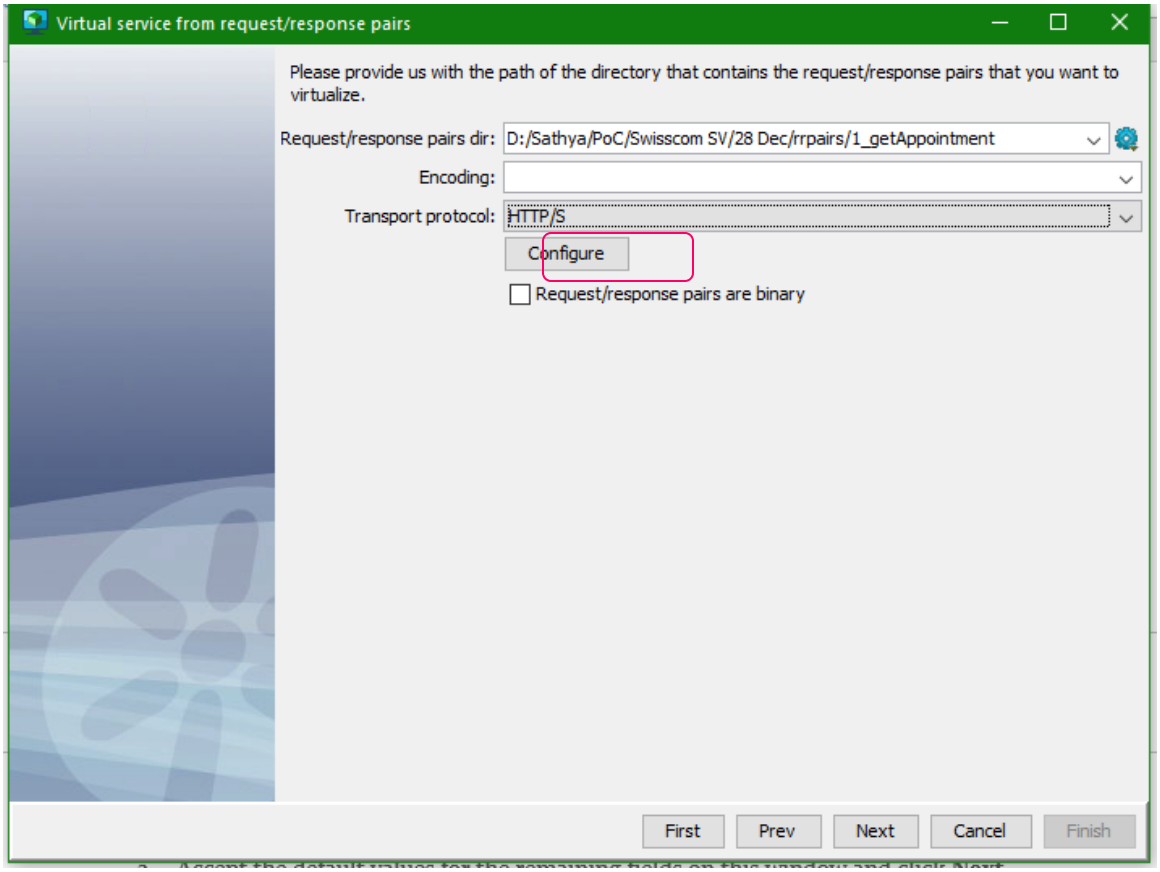
9. Provide name for VSI and VSM & click Next button



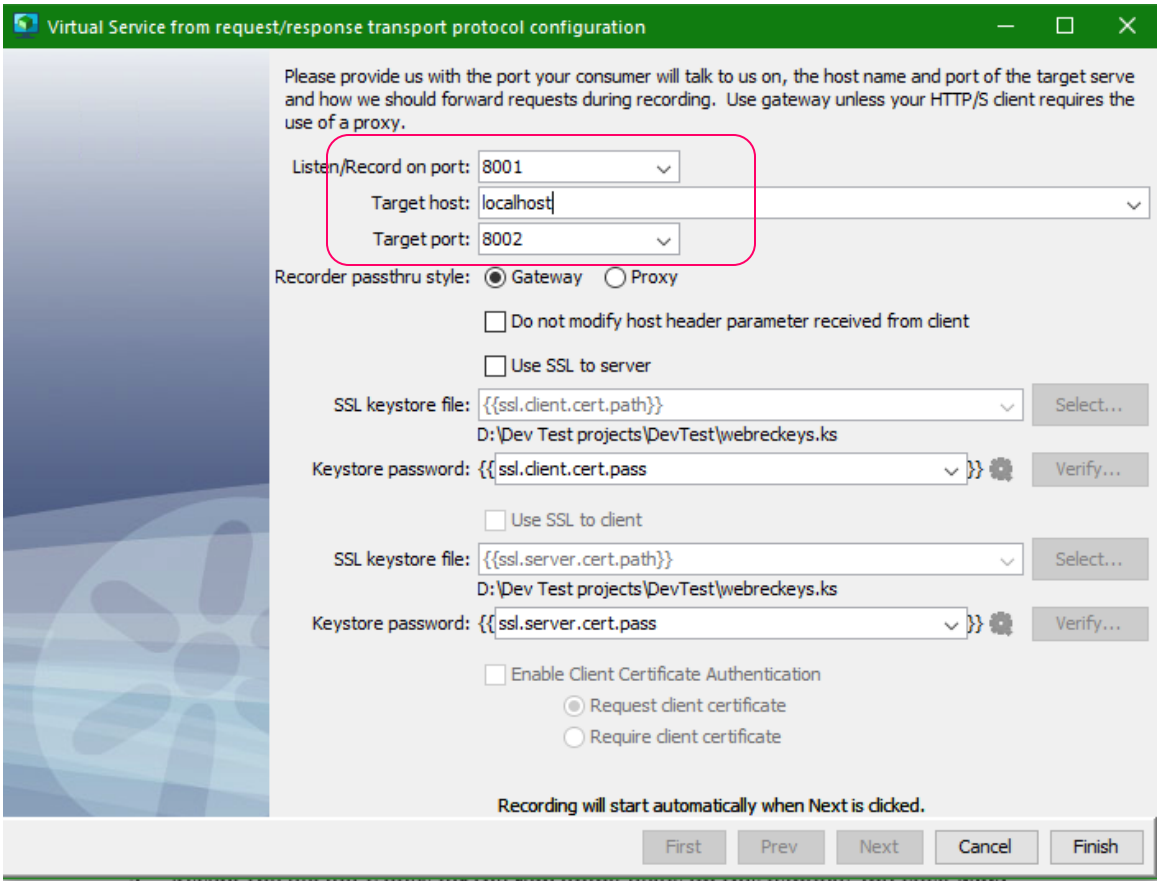
10. Browse the file system from the directory that contains your RR pairs and select corresponding Transport protocol from the drop down list



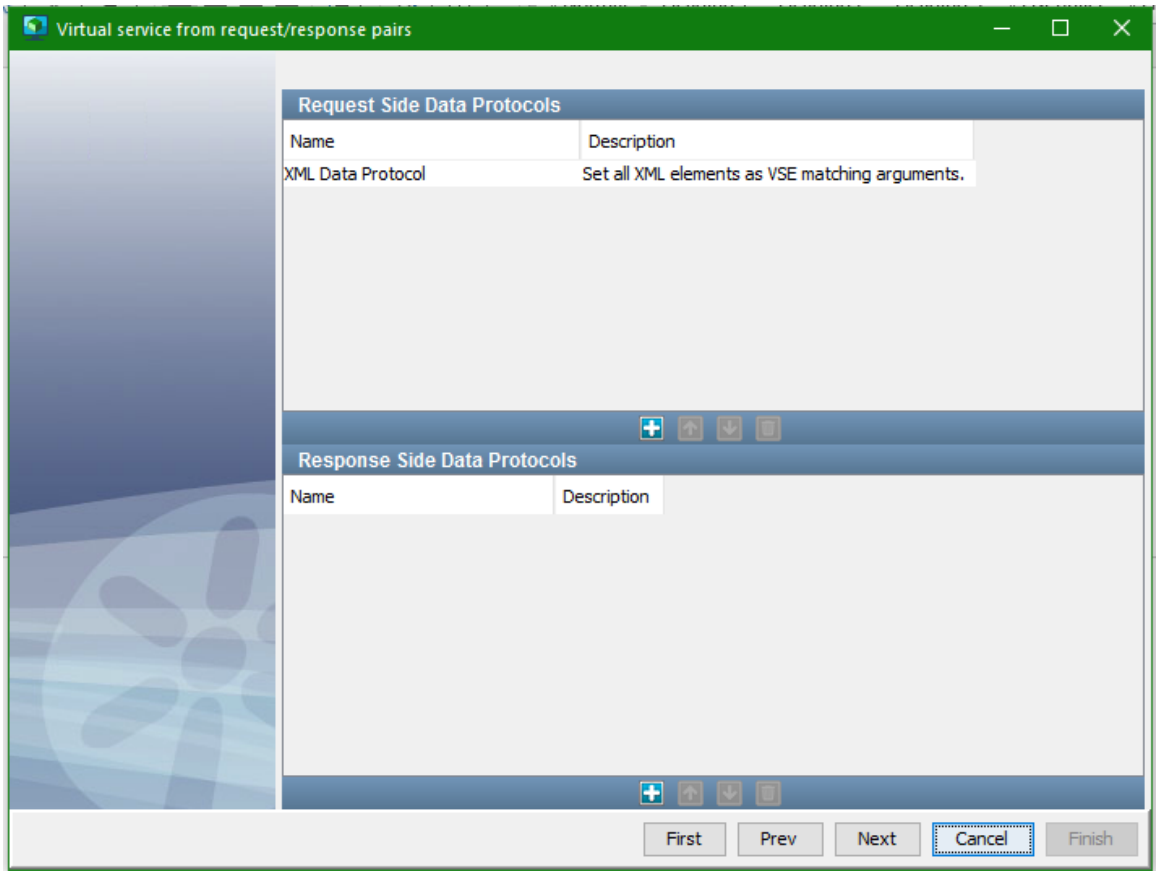
11. Click Configure button



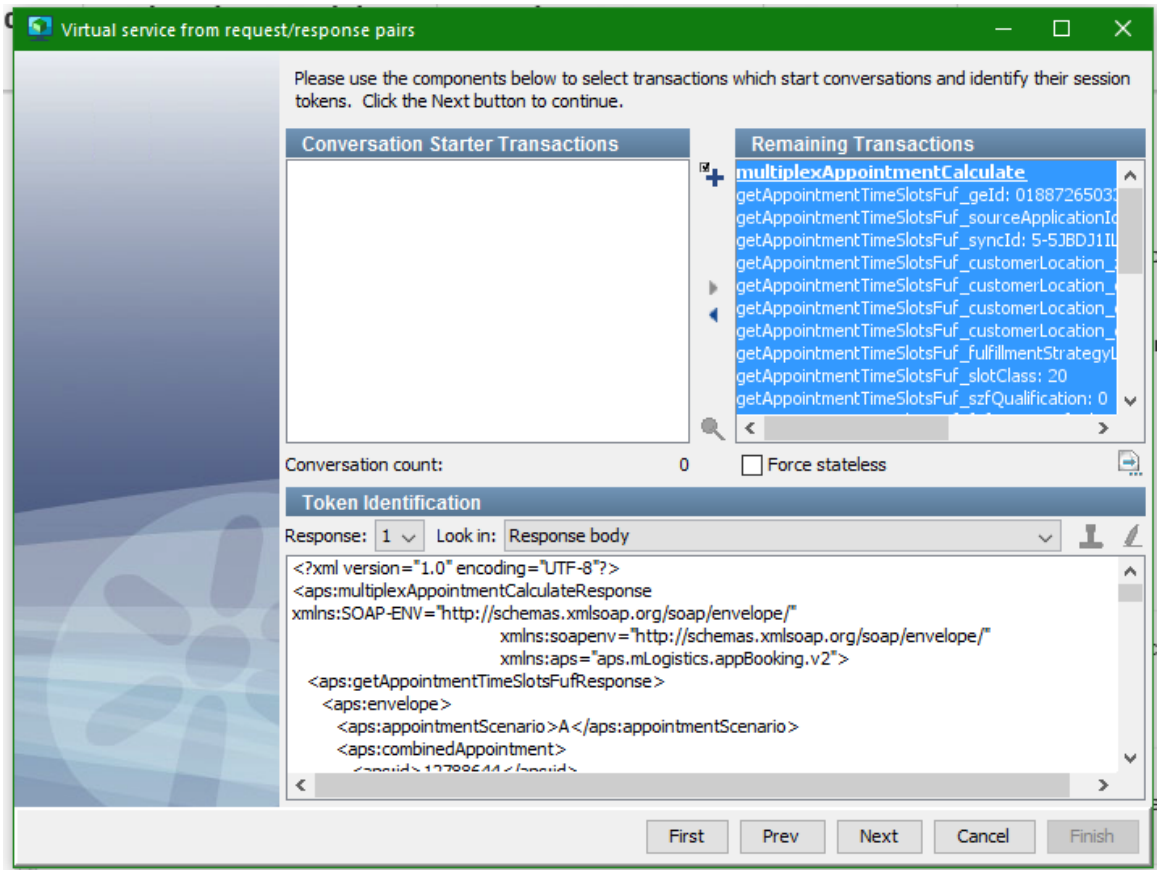
12. Provide necessary details (Listen/Record on Port, Target Host & Target Port) & click Finish and then Next buttons



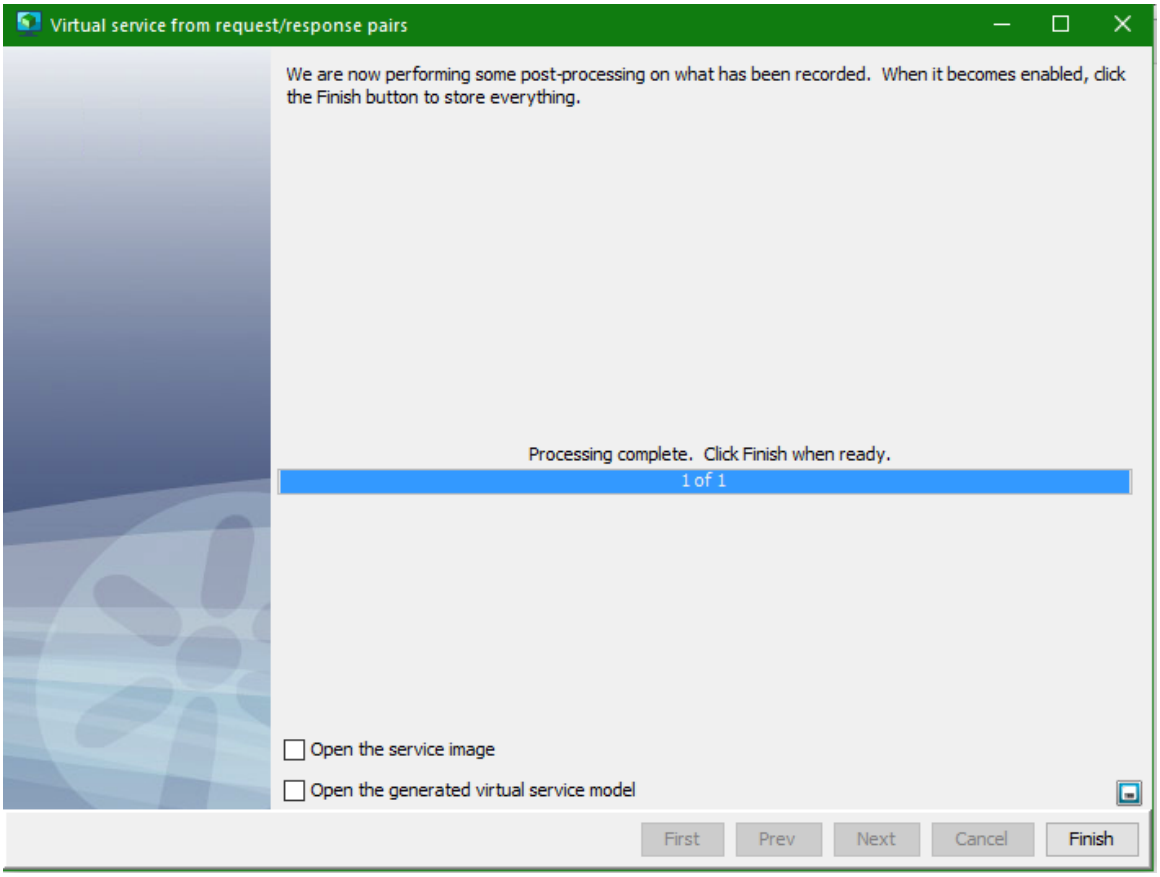
13. Appropriate data protocols for the request and response will be defaulted



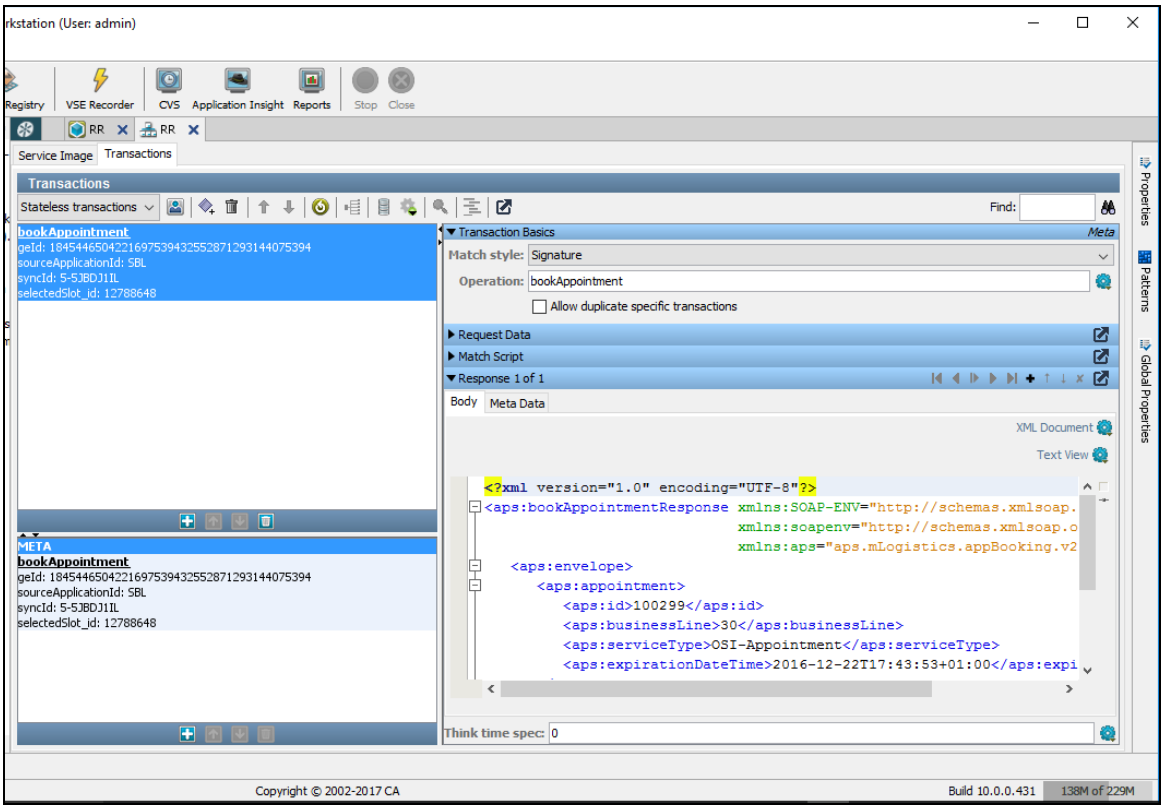
14. Click Next and Next again



15. Click Finish button



16. After processing of the VSM & VSI, you can open the service image and the virtual service model

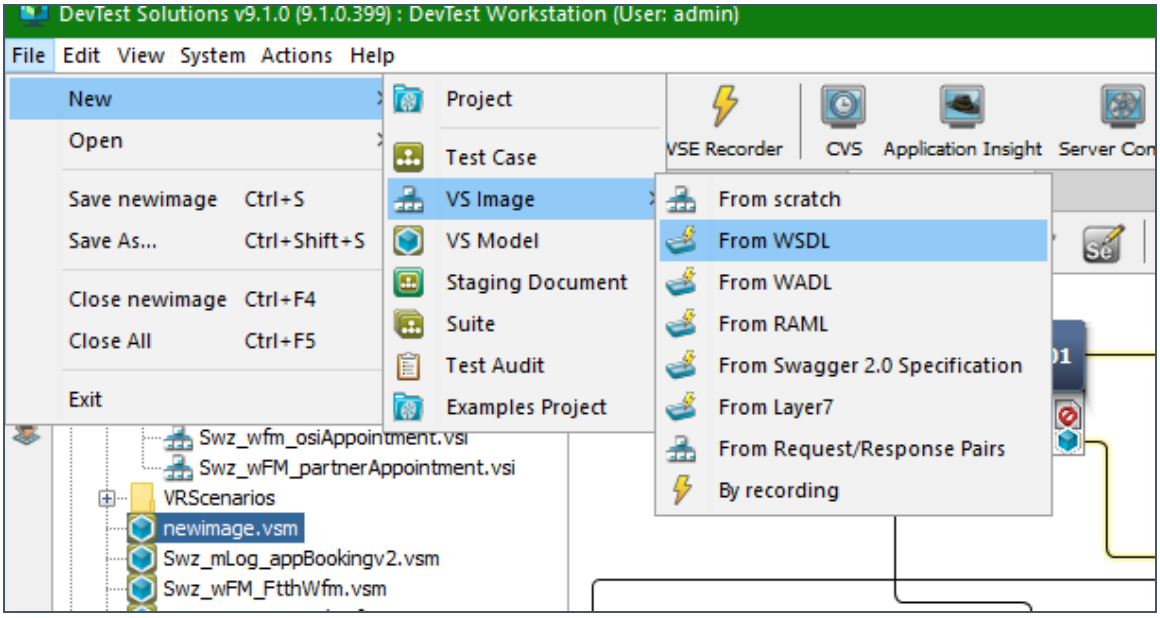


17. You can execute the VSM using SoapUI by copying the Request and executing it with correct port and base path that are displayed in VSM

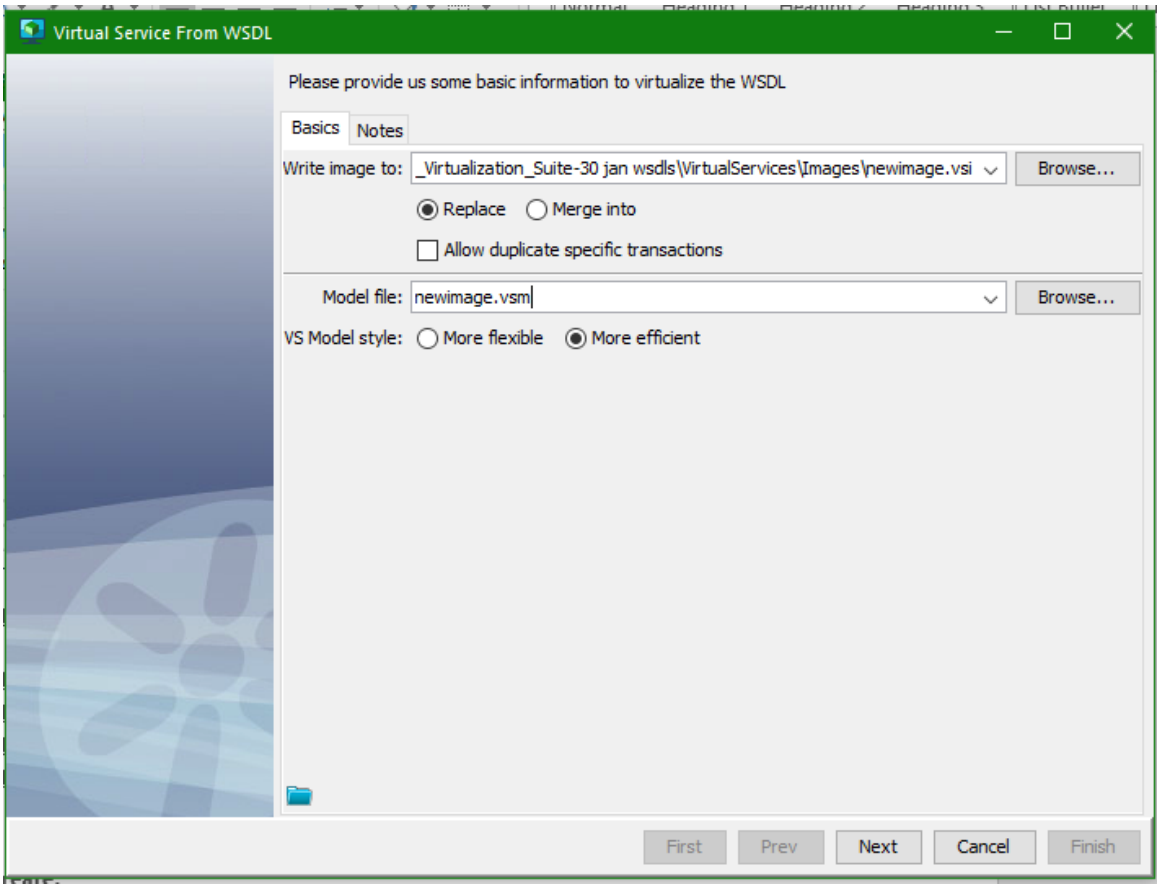
13. Creation of Virtual Service using WSDL's

WSDL is an XML-based protocol. DevTest Portal lets you create a virtual service from one or more WSDL documents. Assume that you want to virtualize a service that is in the early stages of development. You might not be able to create the virtual service by recording. However, if you have access to a WSDL document that describes the service, you can use this approach.

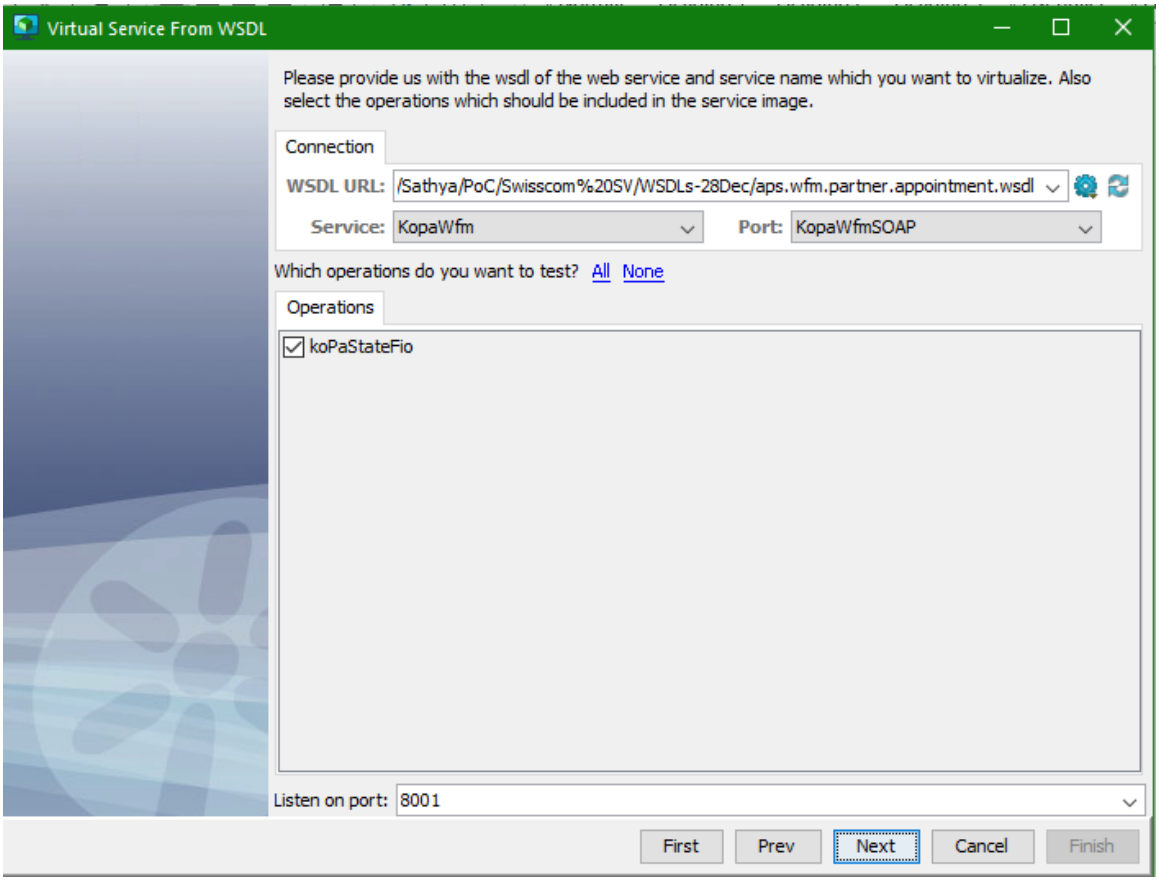
1. Select File > New > VS Image > From WSDL



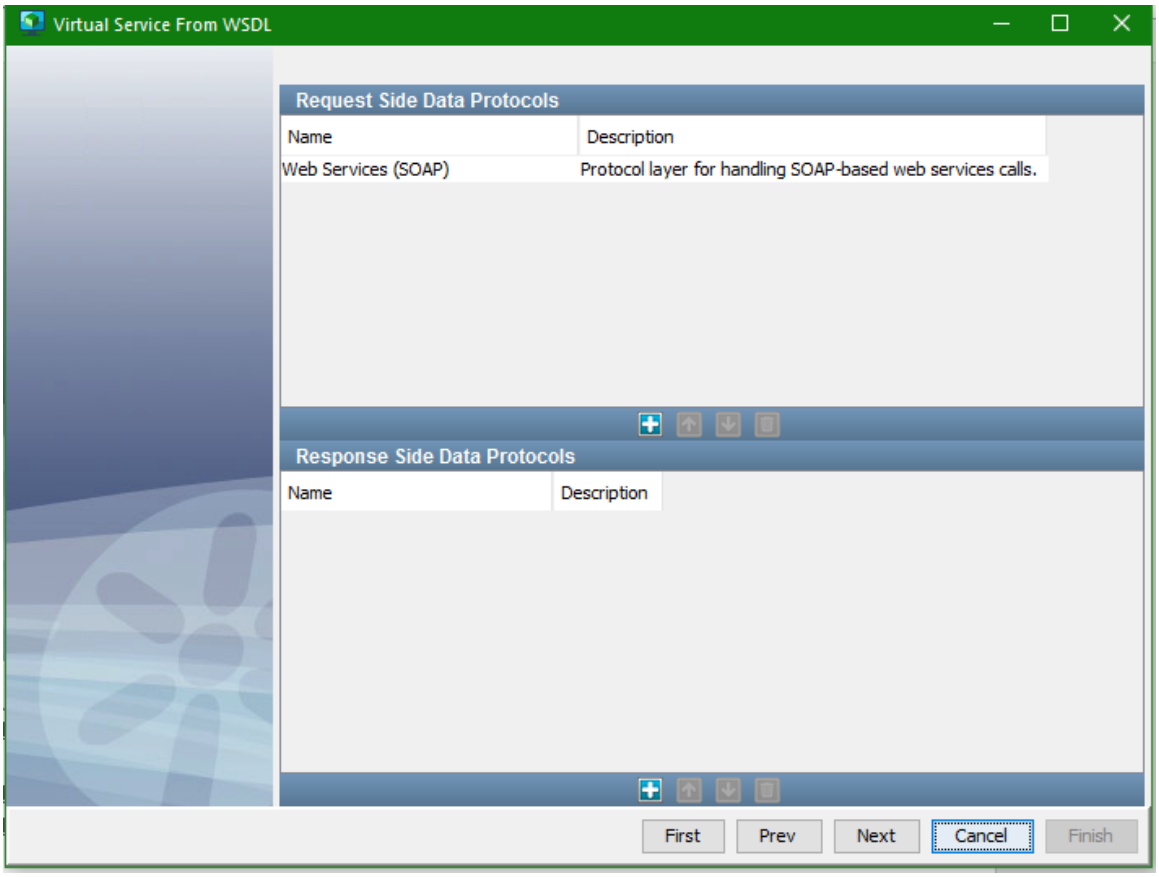
2. Provide name for VSI and VSM and click Next button



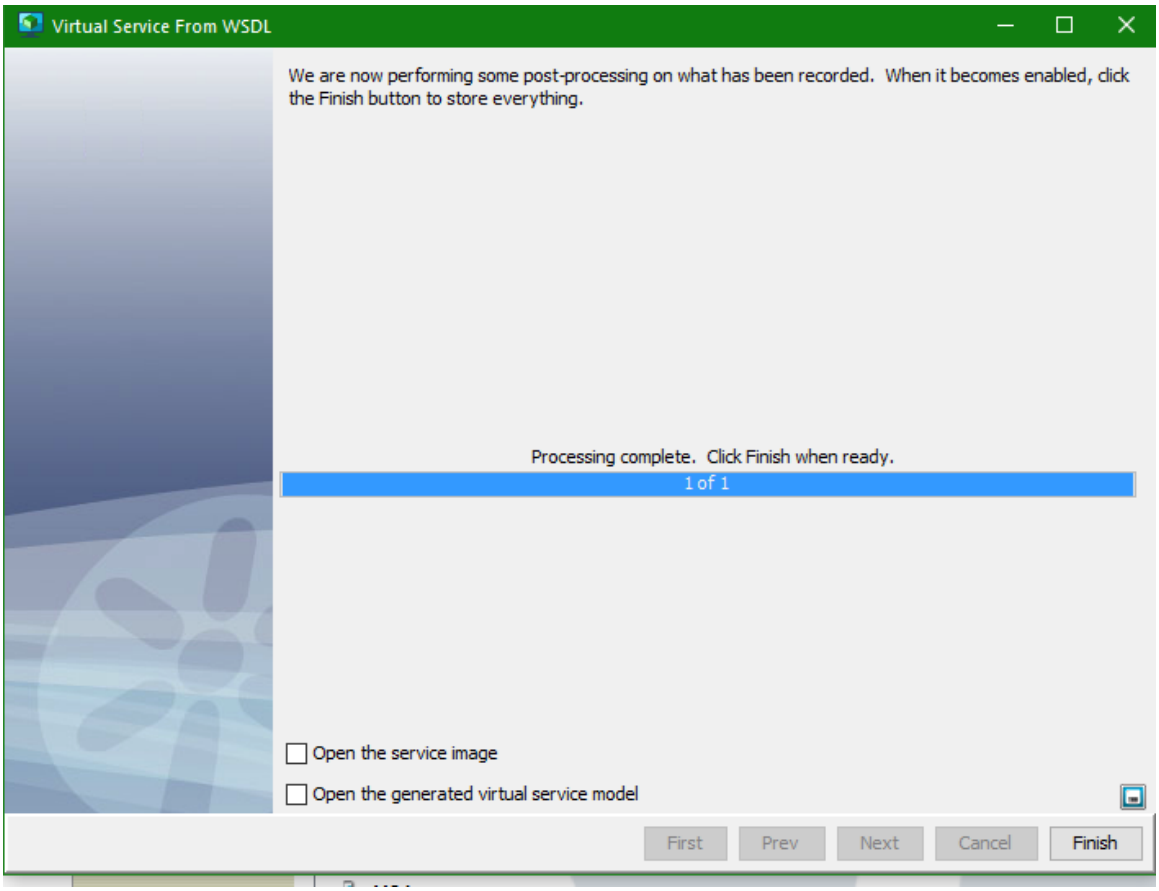
3. Select the path of WSDL file and change the listen port if required and click Next button



4. Click Next button and select corresponding Data Protocol if value is not defaulted by DevTest and click Next button



5. Click Next and Finish button



6. You can execute the VSM using SoapUI by copying the Request and executing it with correct port and base path that we mentioned in VSM
7. Sample files:

Sample files are embedded here:

14. REST API Virtualization

REST SERVICES USING HTTP PROTOCOL:

The REST data protocol handler analyses HTTP requests that follow the REST architectural style. The data protocol identifies the dynamic parts of the URI strings. The result is a set of rules. Virtual service environment (VSE) uses the rules to virtualize HTTP requests that invoke the same operations. We can include the HTTP requests in live traffic or in request/response pairs. We must use the HTTP/S transport protocol for request/response pairs. The REST data protocol handler supports all the HTTP methods/verbs: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT, and PATCH.

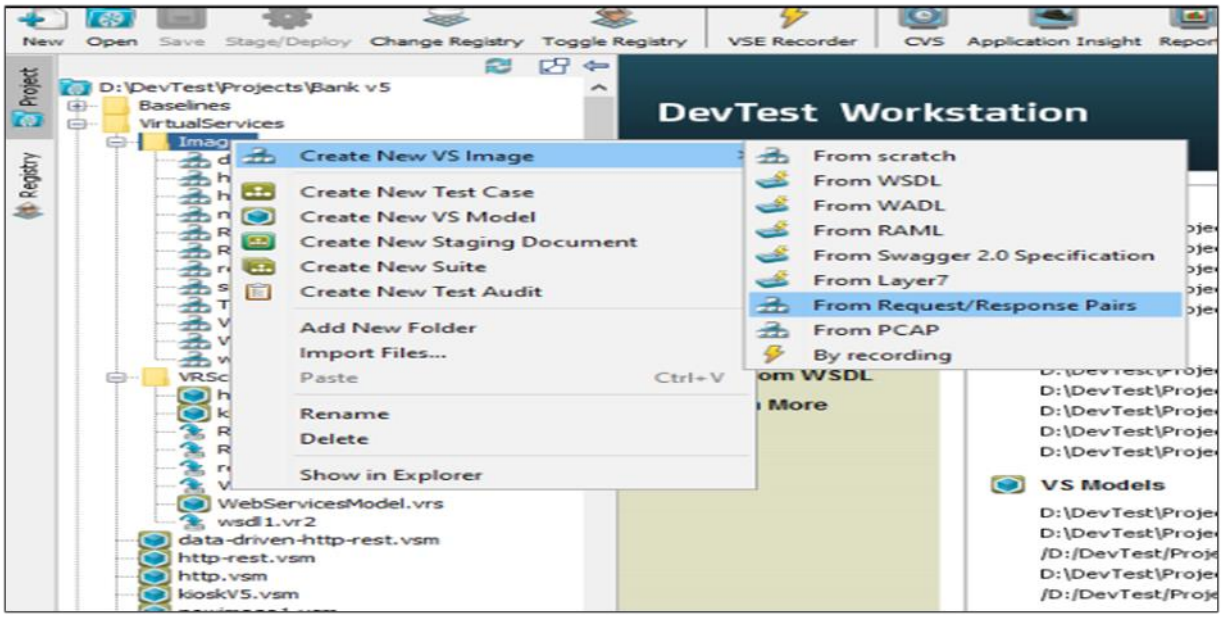
- **Request/Response Pair Format:** The format of request/response pairs for the REST data protocol
- **Request**: The request file must include a valid HTTP header. The URL is the first line of the header. The other header lines are optional

  The format of the URL line is: <METHOD><a space character><REST API path><space><HTTP-VERSION>
- **Response**: The response file contains an HTTP response code. The file can contain a header and a body. The response code must be the first line in the file, in the following format: <HTTP-VERSION><a space character><HTTP-RESPONSE-CODE>

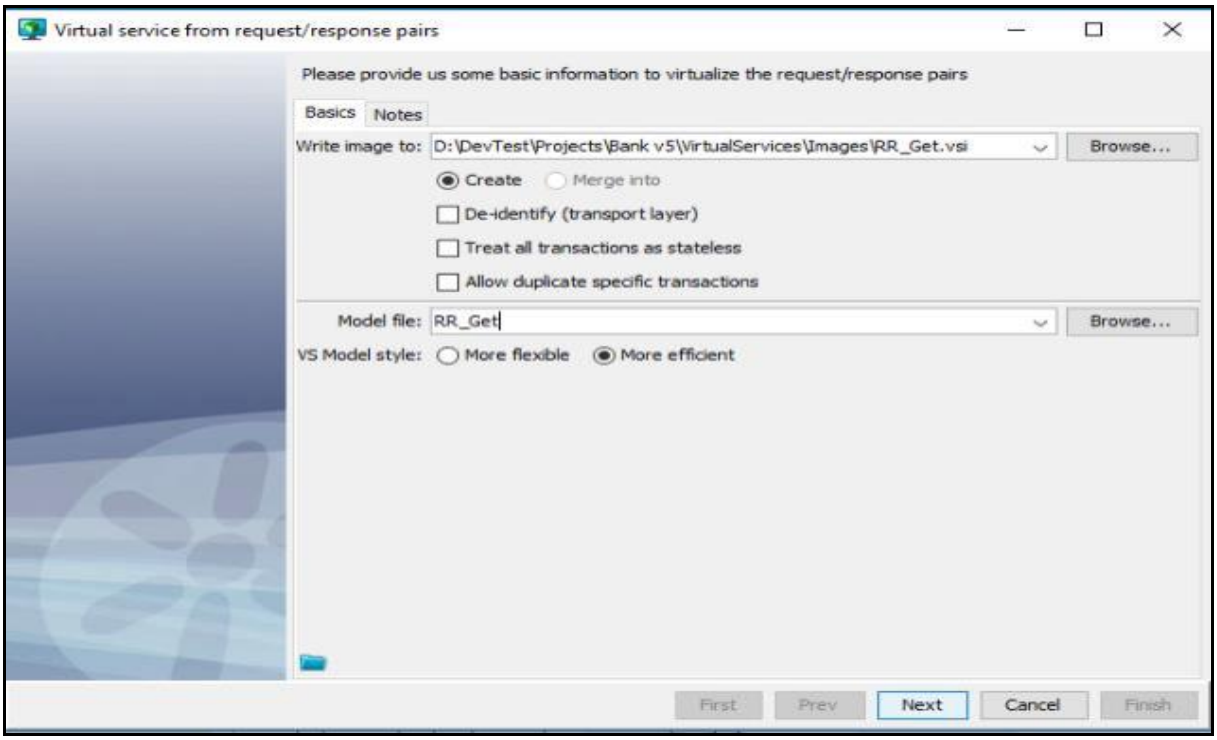
14.1 Request and Response Pair Using Rest Get Method

GET method is used to read or retrieve a representation of a resource. GET returns a representation in XML or JSON.

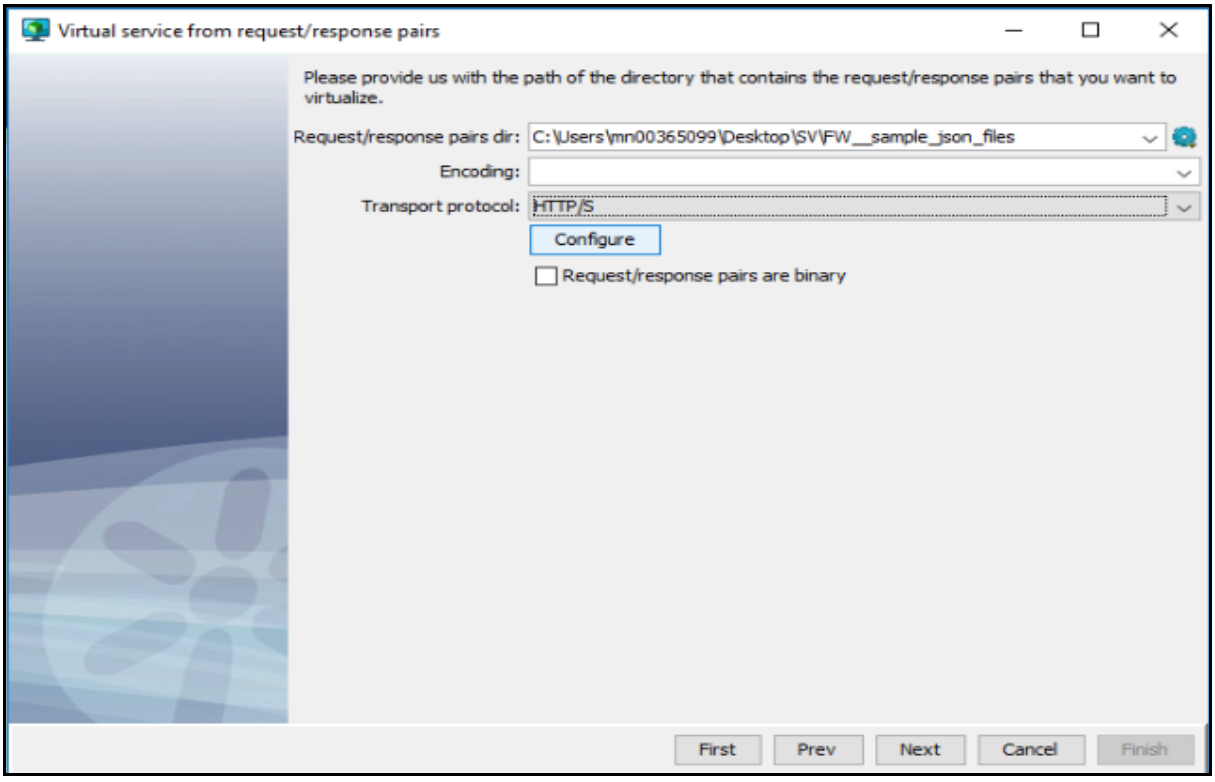
1. Select Projects > Images > Create New VS Image > From Request/Response Pairs



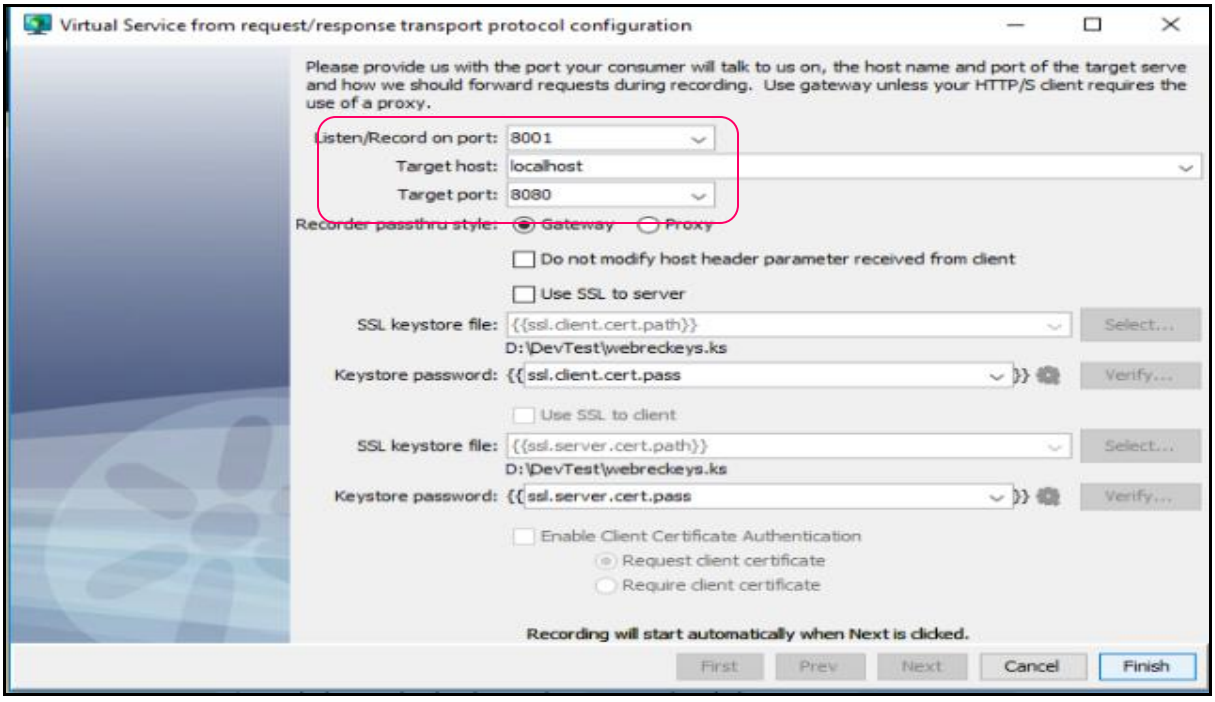
2. Enter Service Image name and Model name, click Next button



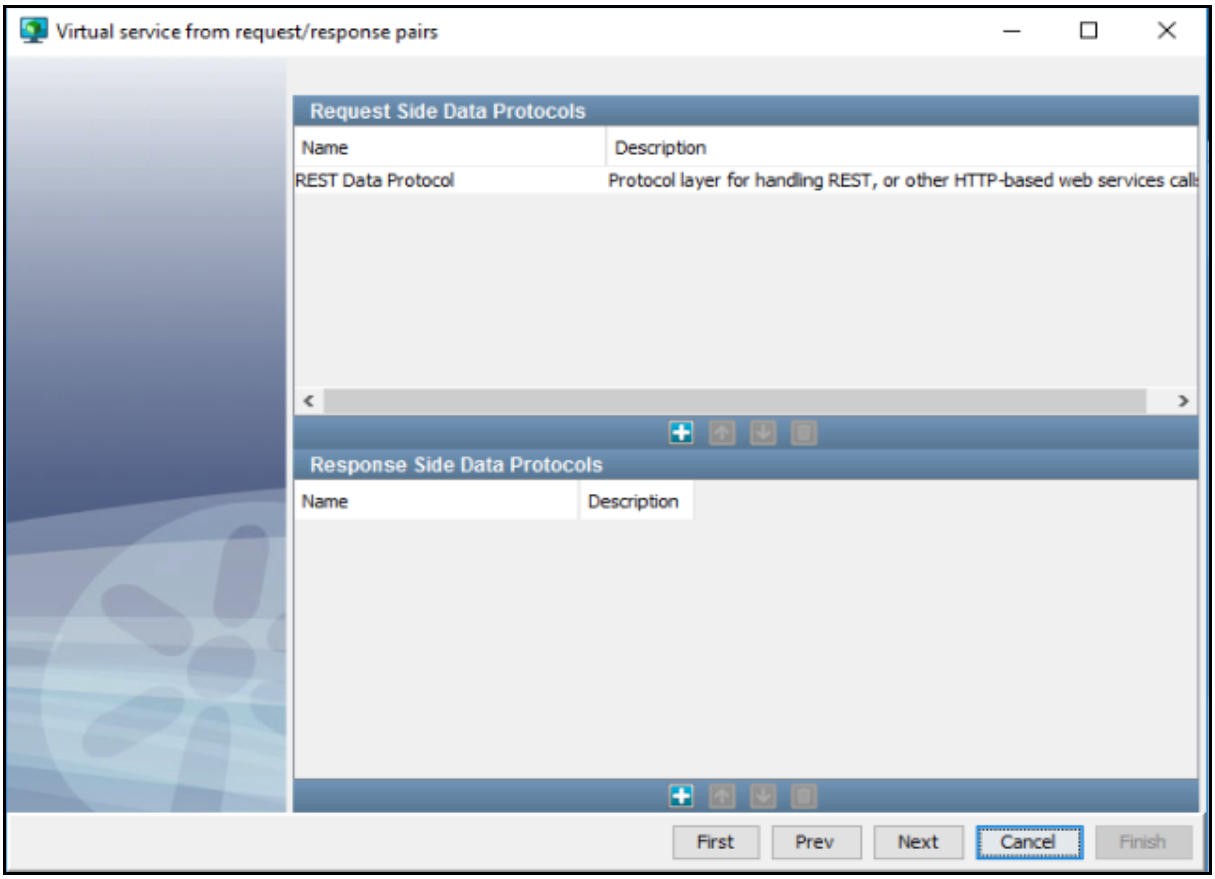
3. Browse the path for request/response pairs & select Transport protocol from the drop down list. Click Configure button



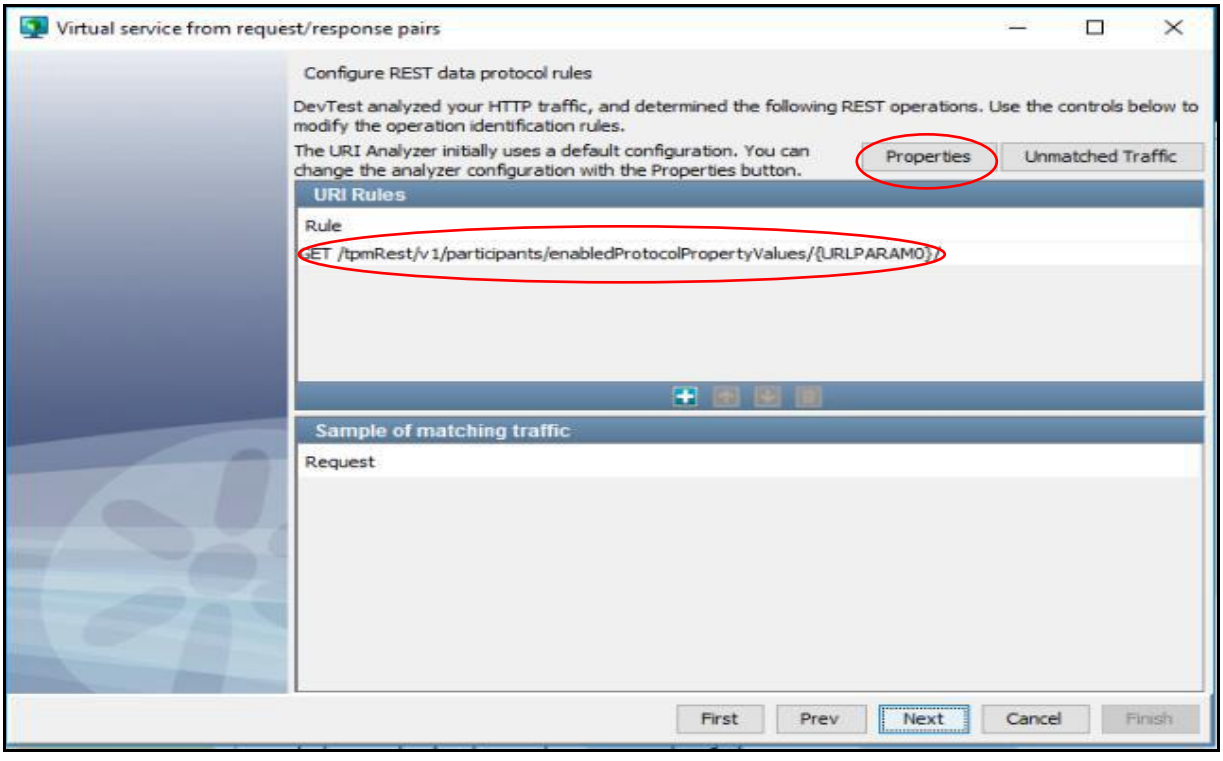
4. Provide details for Listen port, Target host & Target port and click Finish button



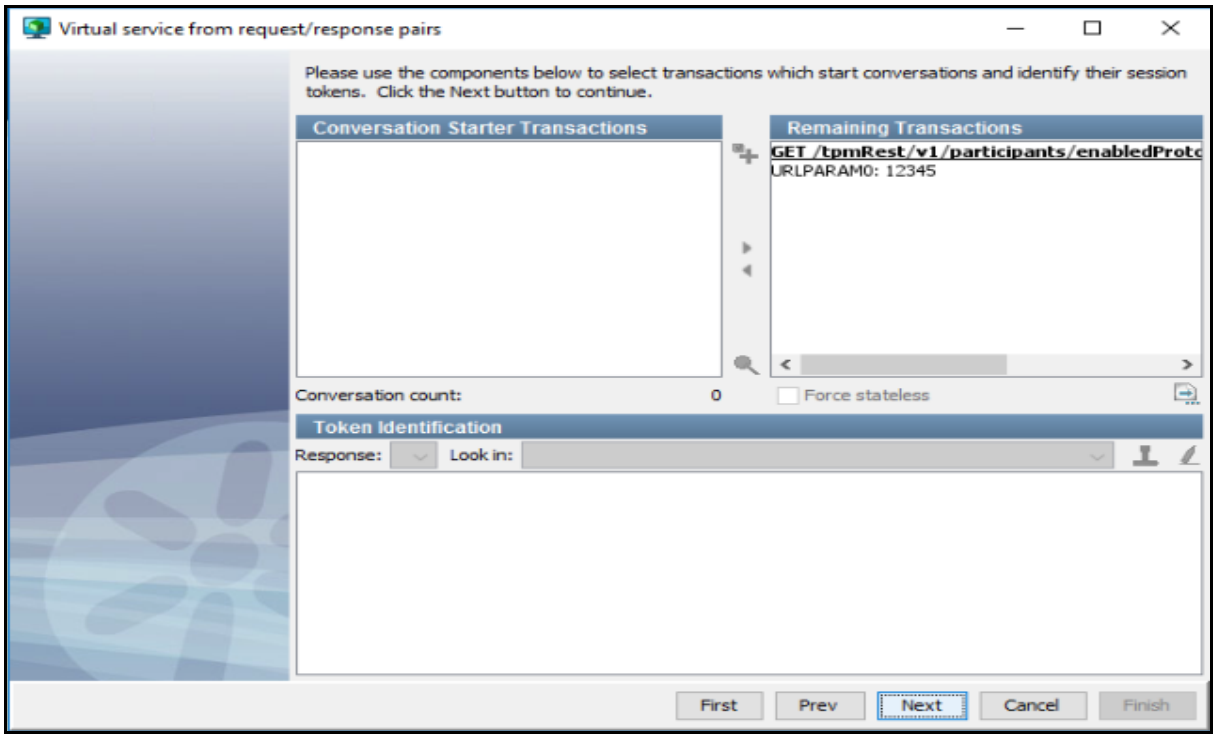
5. Appropriate data protocols for the request and response will be defaulted and click Next button



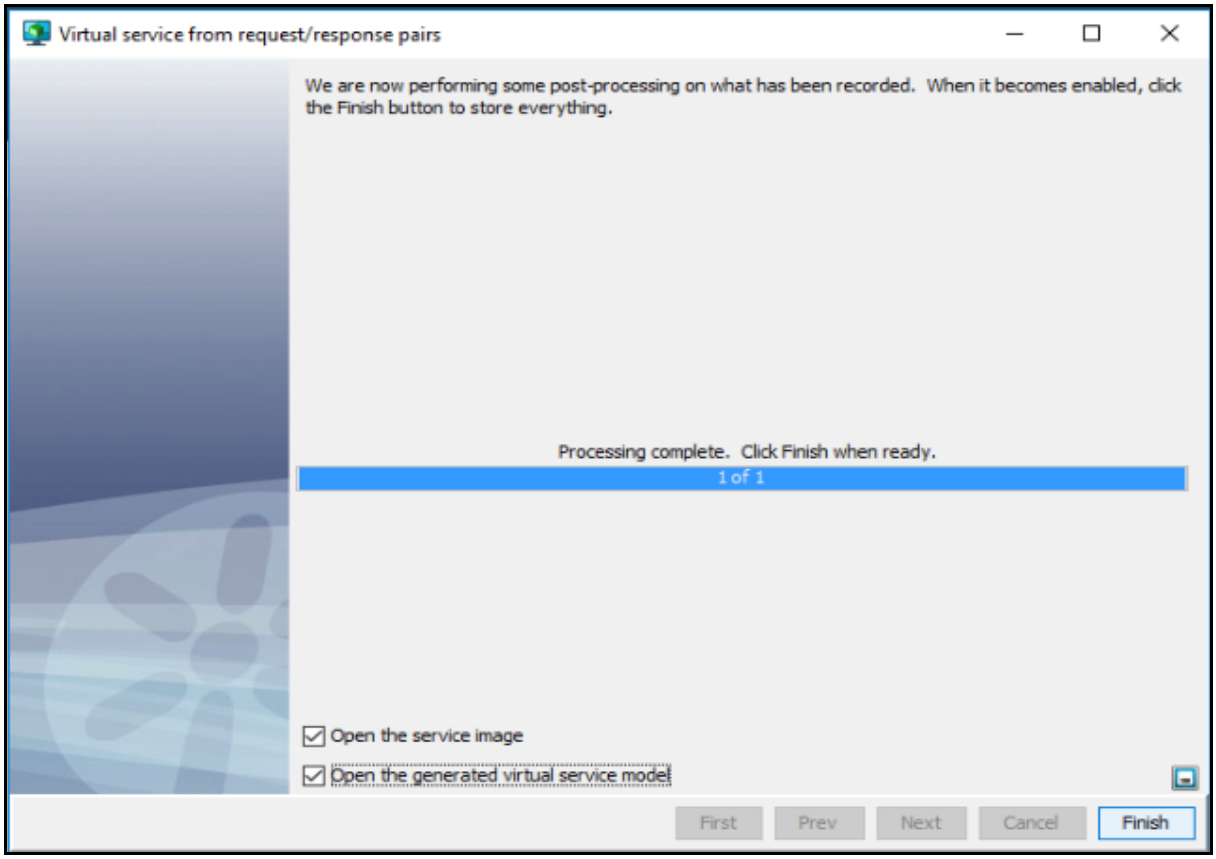
6. Appropriate URI rule will get generated. Rule can be modified by going to Properties option. Click Next button.



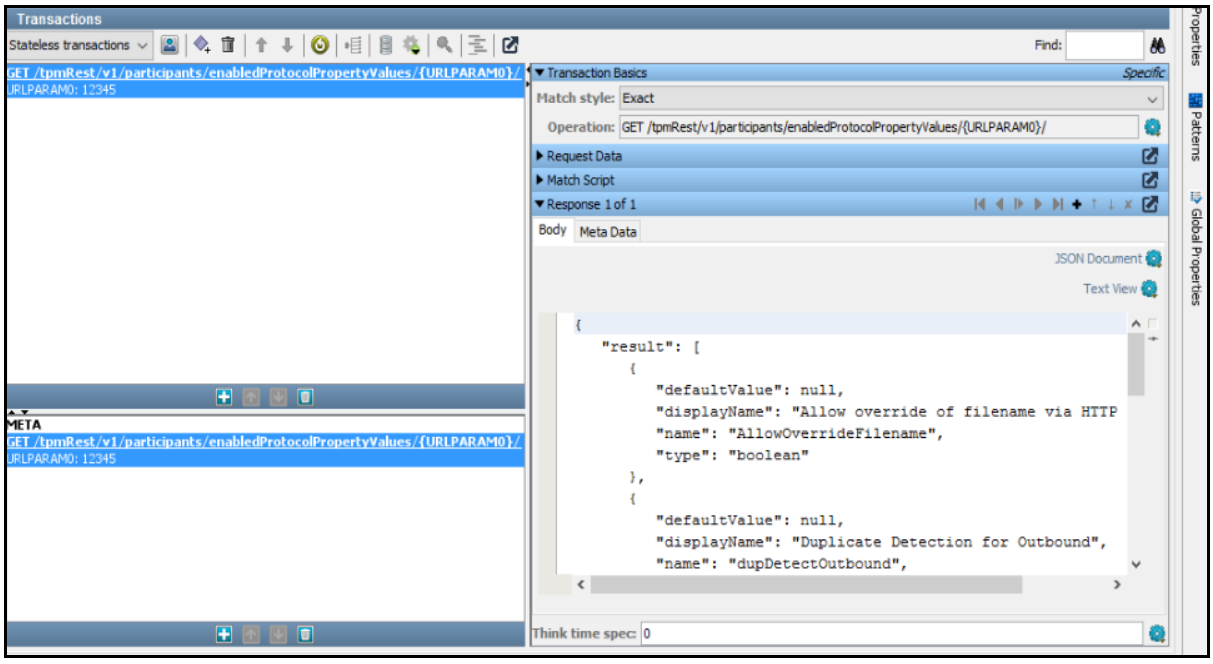
7. Click Next button



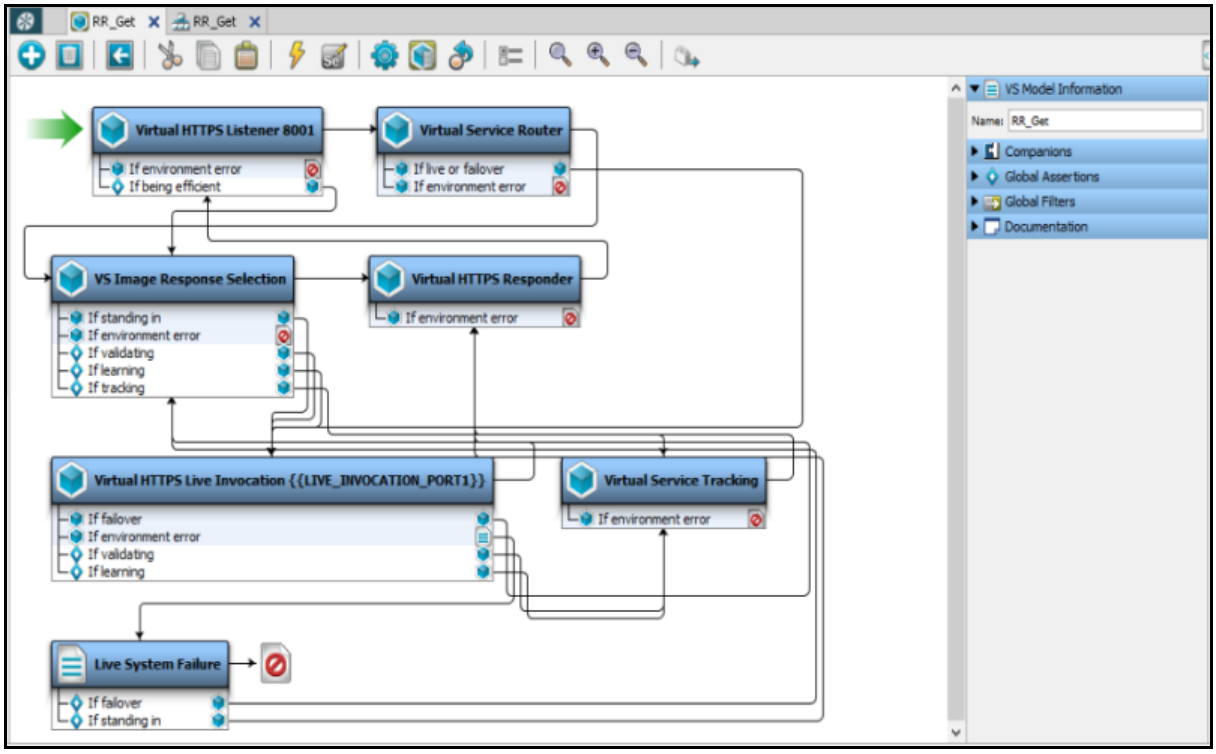
8. Select checkboxes to open VSI and VSM and click Finish button



9. Virtual Service Image created

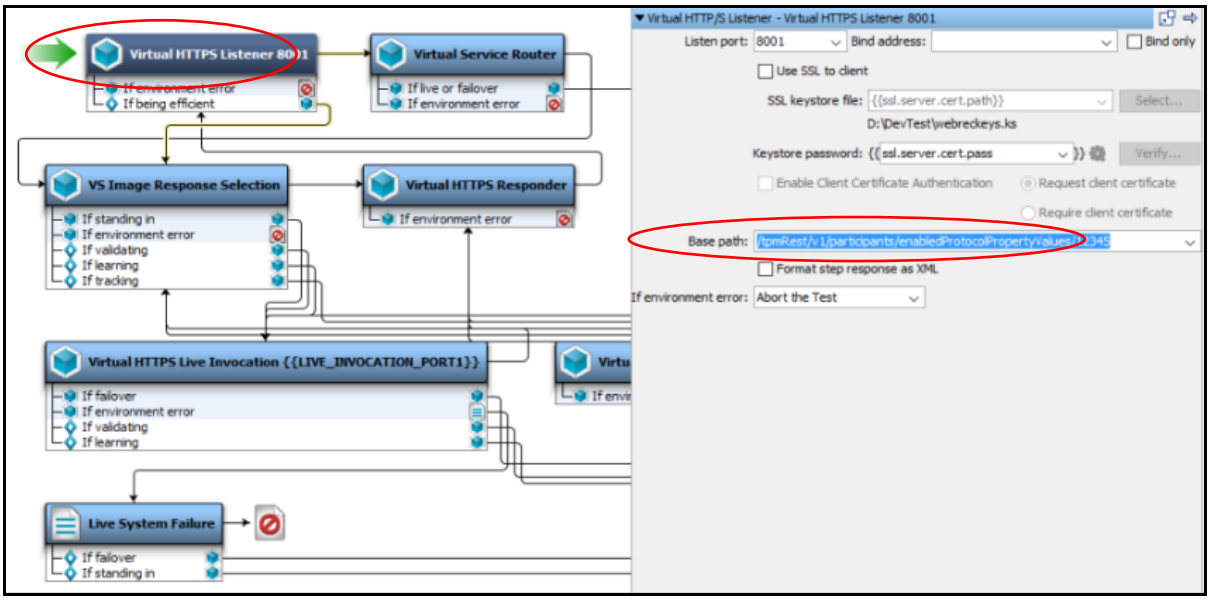


10. Virtual Service Model created

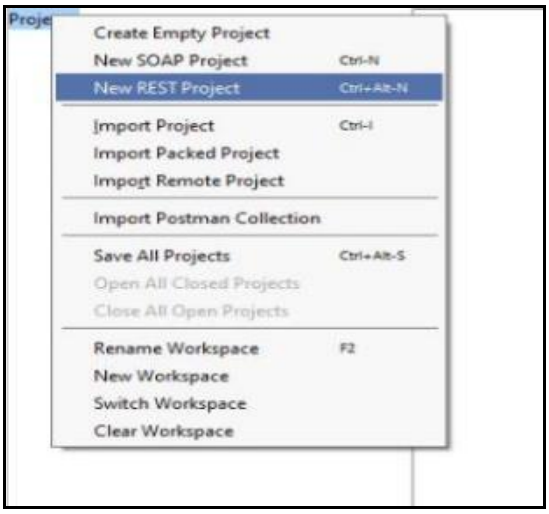


Note: Now we will hit the same request from SOAP UI and test whether the response generated is same as in Virtual Service. Follow steps to execute in SOAP UI

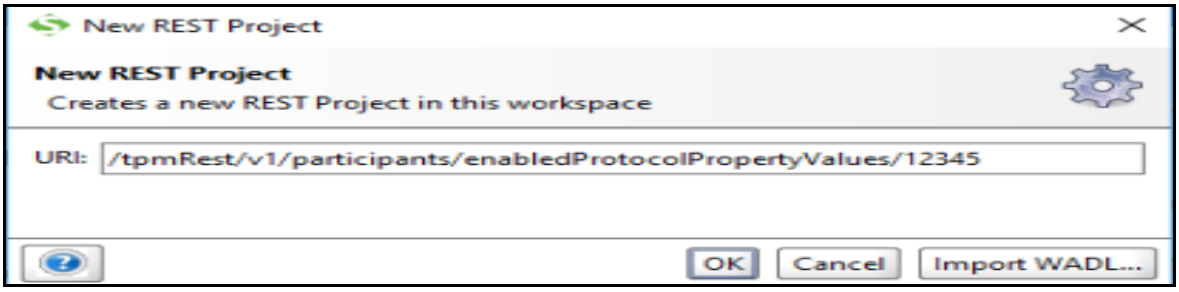
11. Double click the listener port and copy the base path



12. Go to SOAP UI > Projects > New Rest Project



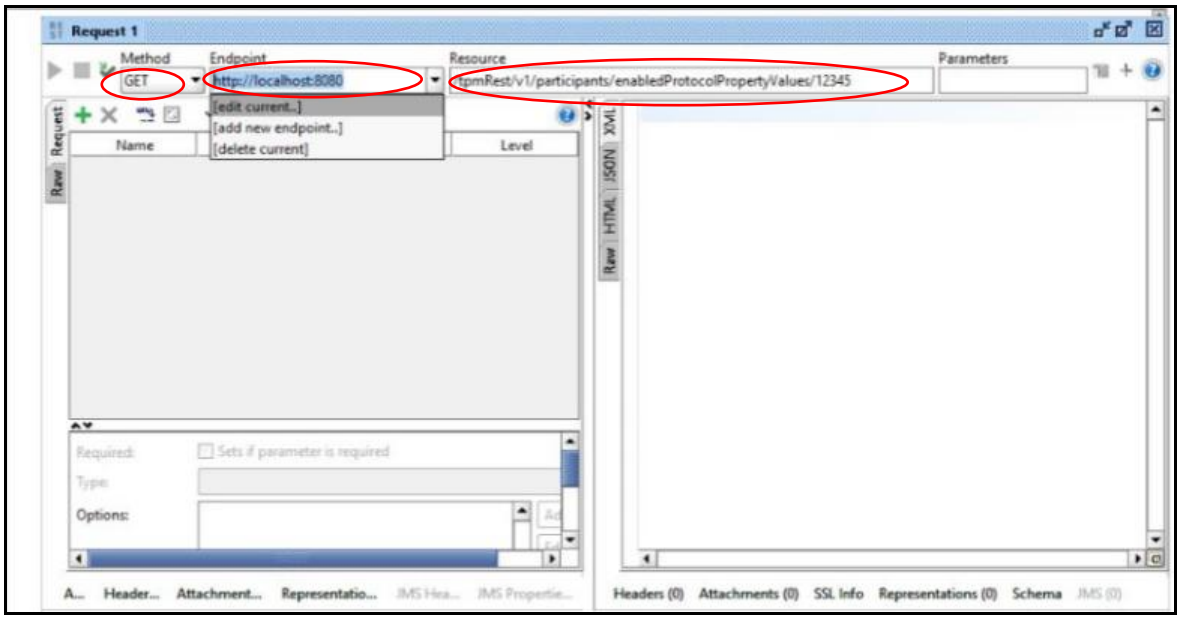
13. Paste the copied base path from VSM and click on OK button



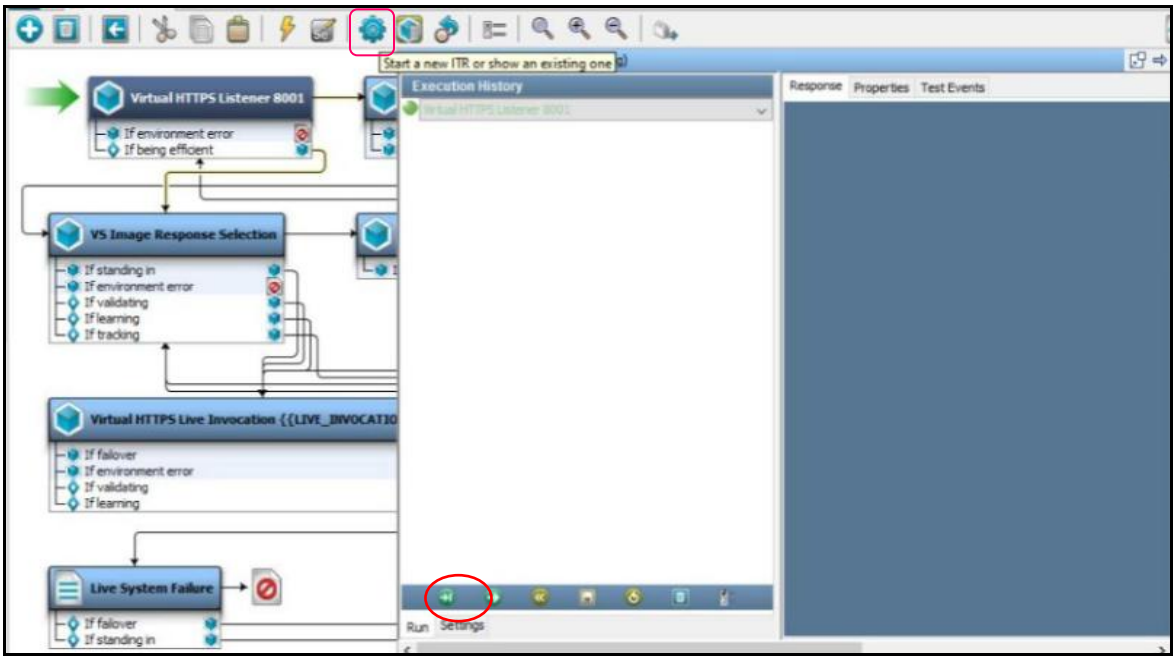
14. Enter the necessary endpoint, method, resource and parameters details

- Endpoint: Transport protocol://target host: listener port.
- Method: Get

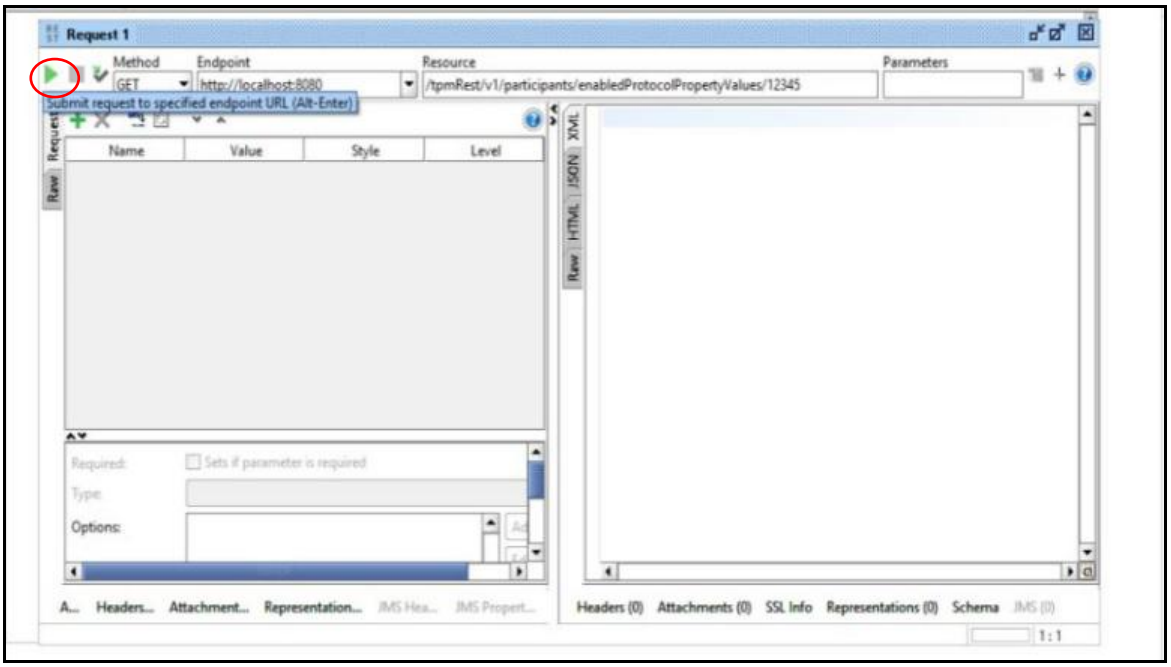
- Resource: Base Path



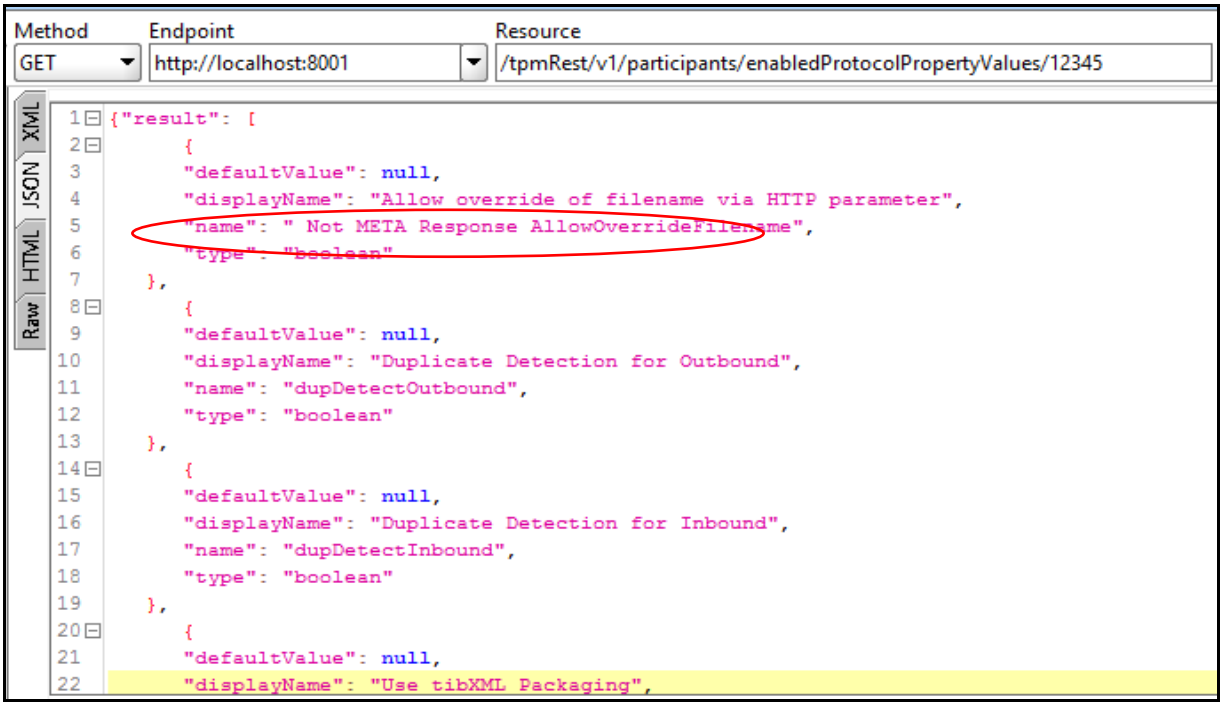
15. Go to VSM and start ITR and click on automatically execute the test option



16. Click on run button in SOAP UI

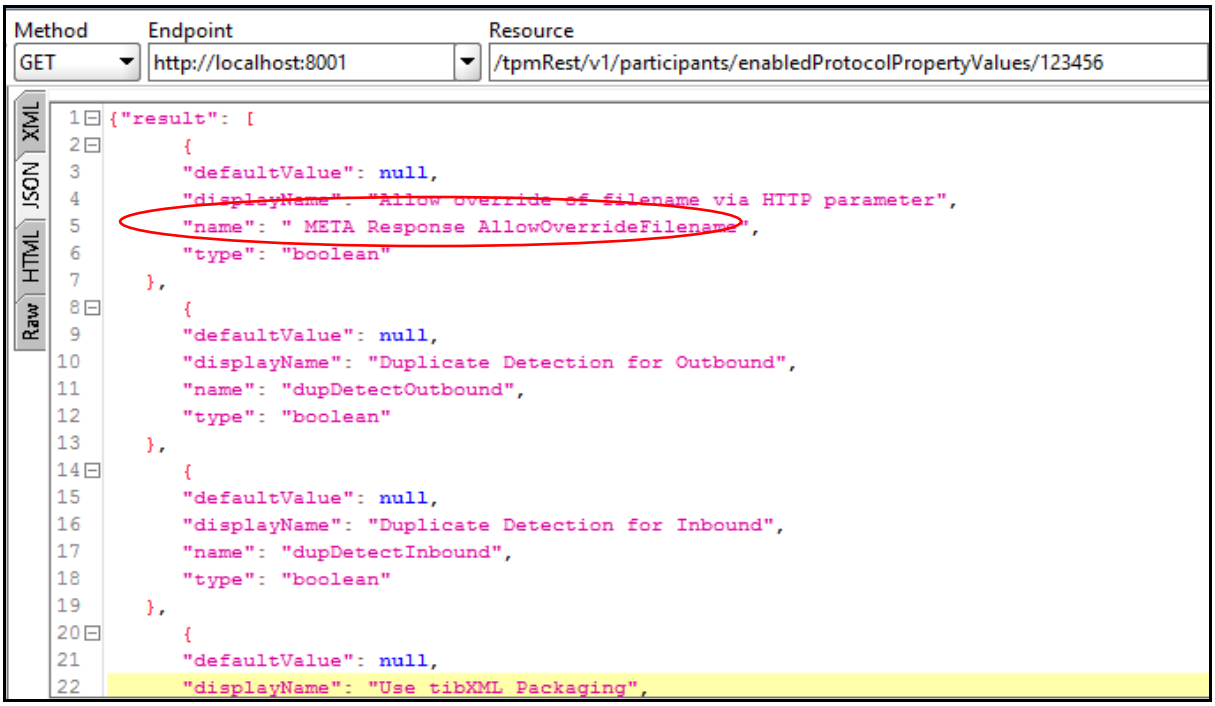


17. SOAP UI will hit the same response as in Virtual Service.

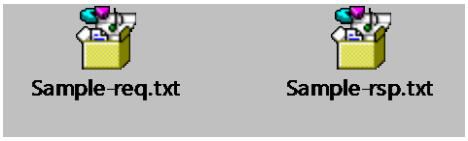


18. SOAP UI will hit the META Response if current request doesn't match the actual request.

E.g.: (Actual Request-12345, Current Request – 123456)



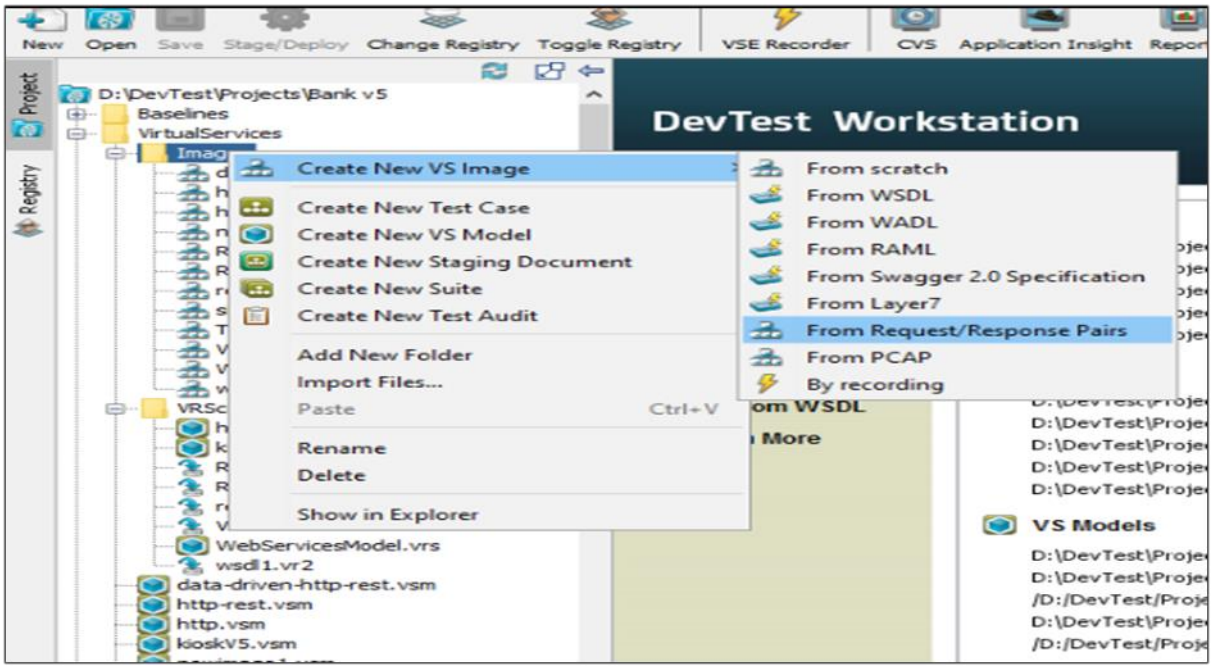
Sample: Request and Response Pair files



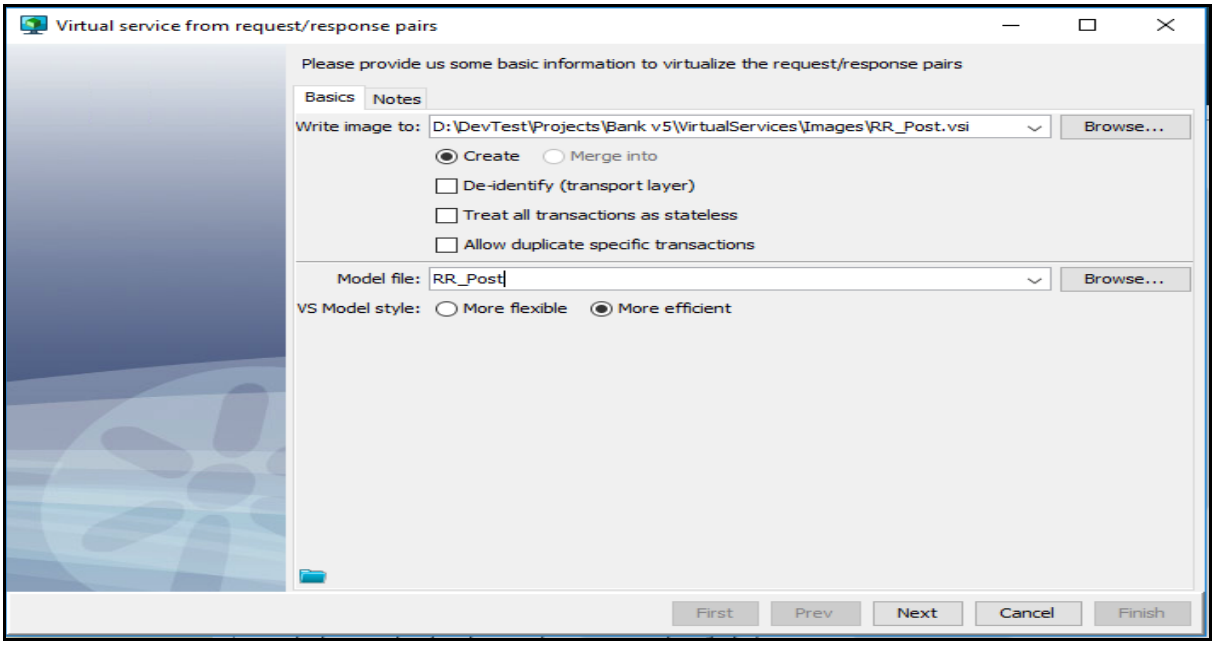
14.2 Request and Response Pair Using Rest Post Method

POST b is most-often utilized to create new resources. In particular, it's used to create subordinate resources. Steps follows:

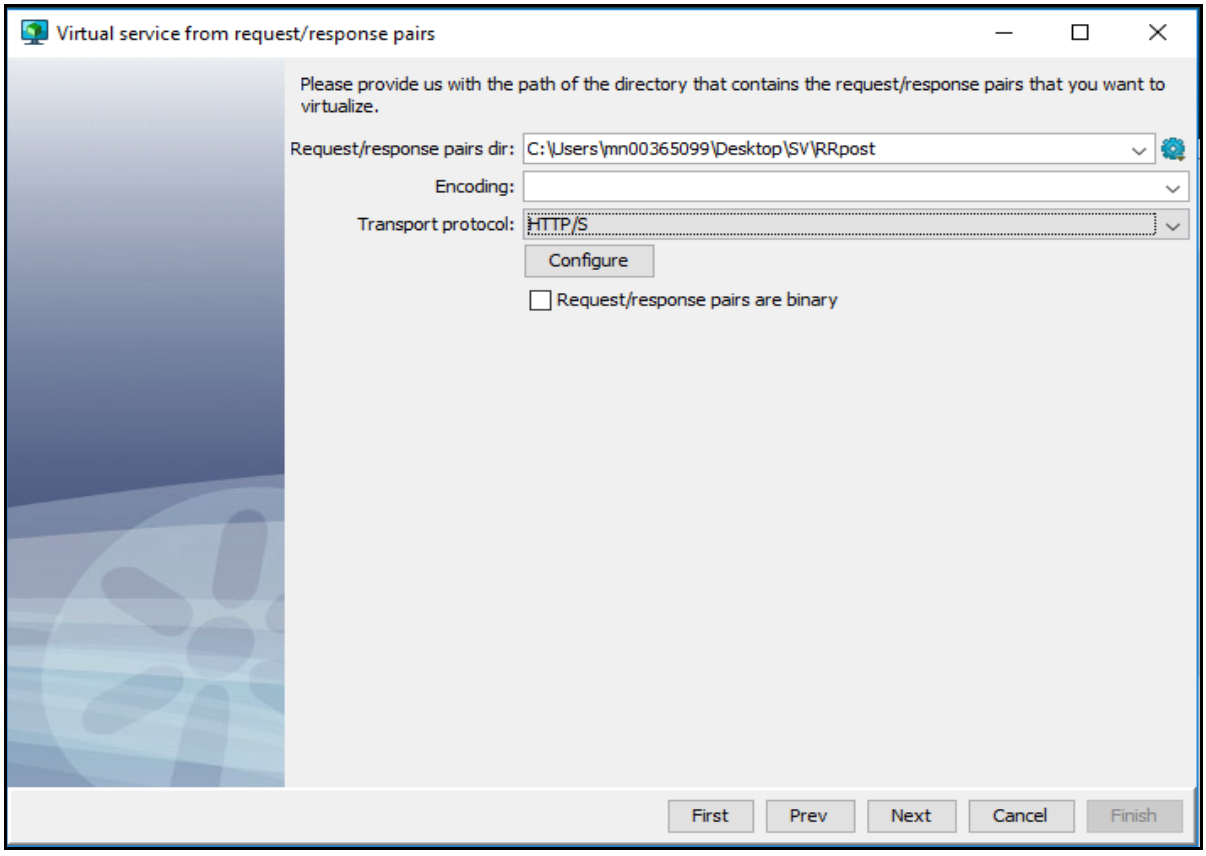
1. Select Projects > Images > Create New VS Image > From Request/Response Pairs



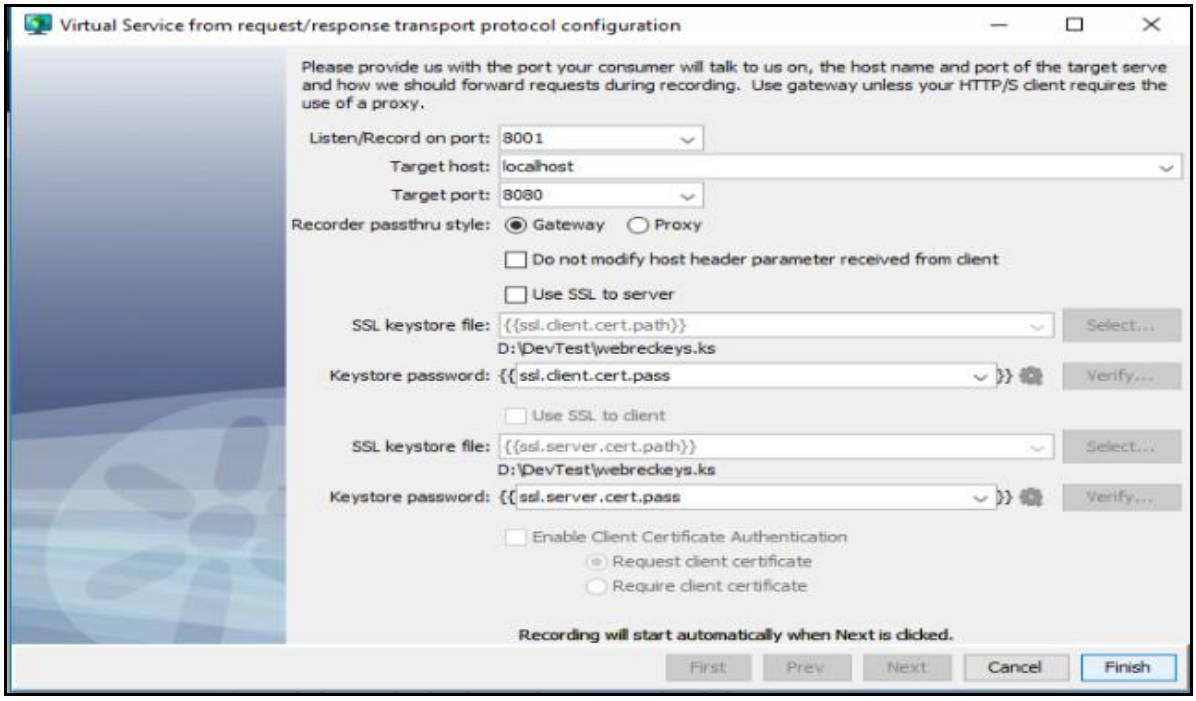
2. Enter Service Image name and Model name, click Next button



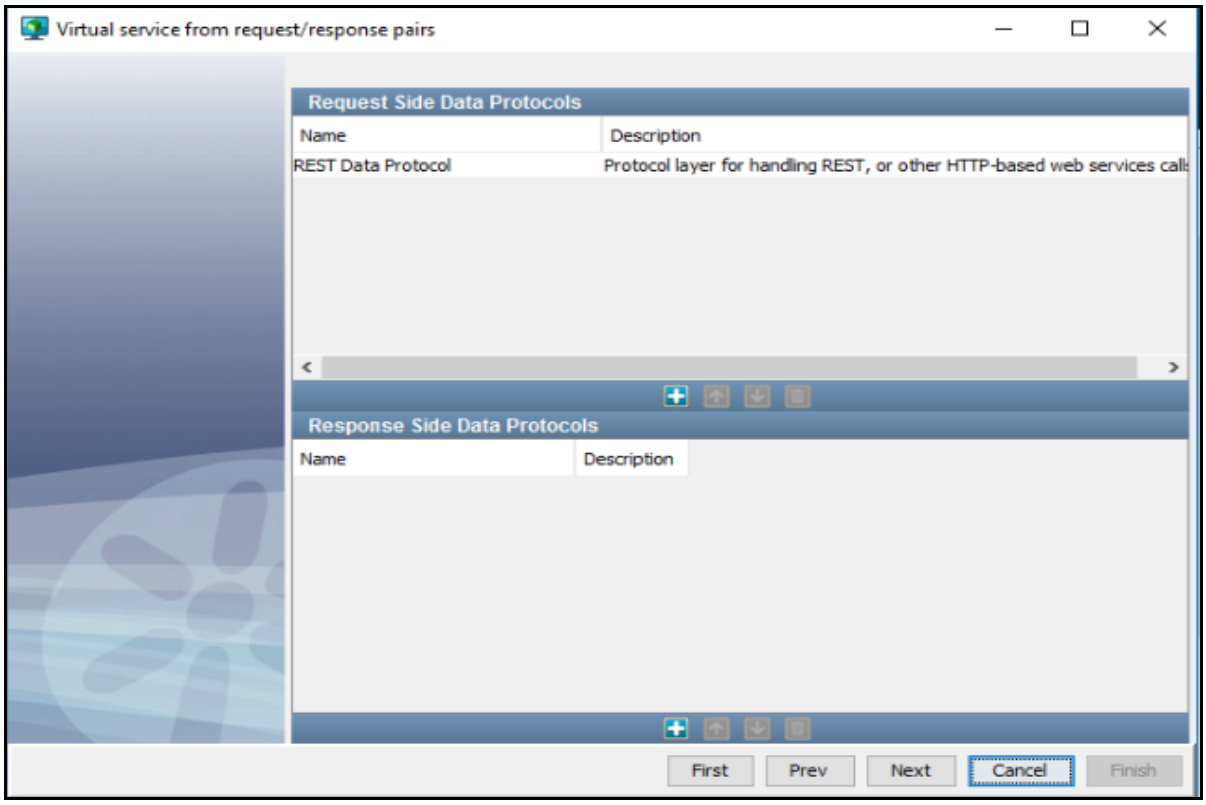
3. Browse the path for request/response pairs & select Transport protocol from the drop down list. Click Configure button



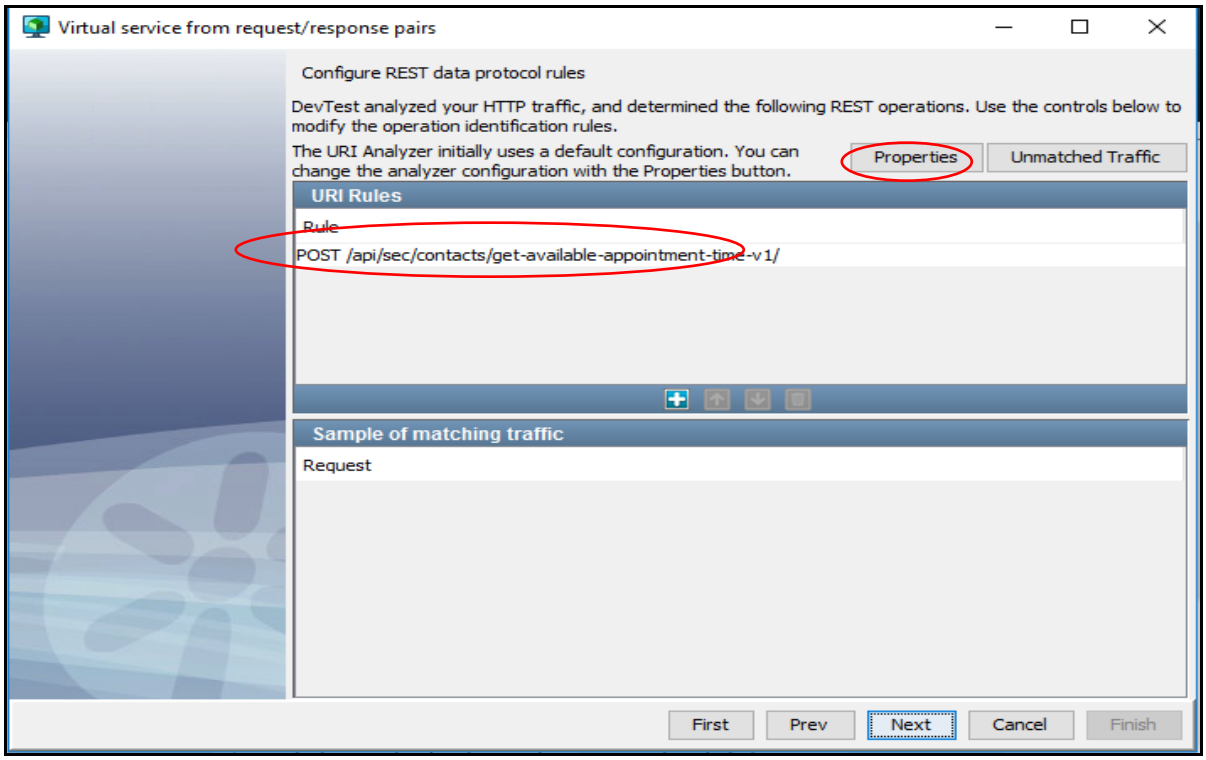
4. Provide details for Listen port, Target host & Target port and click Finish button



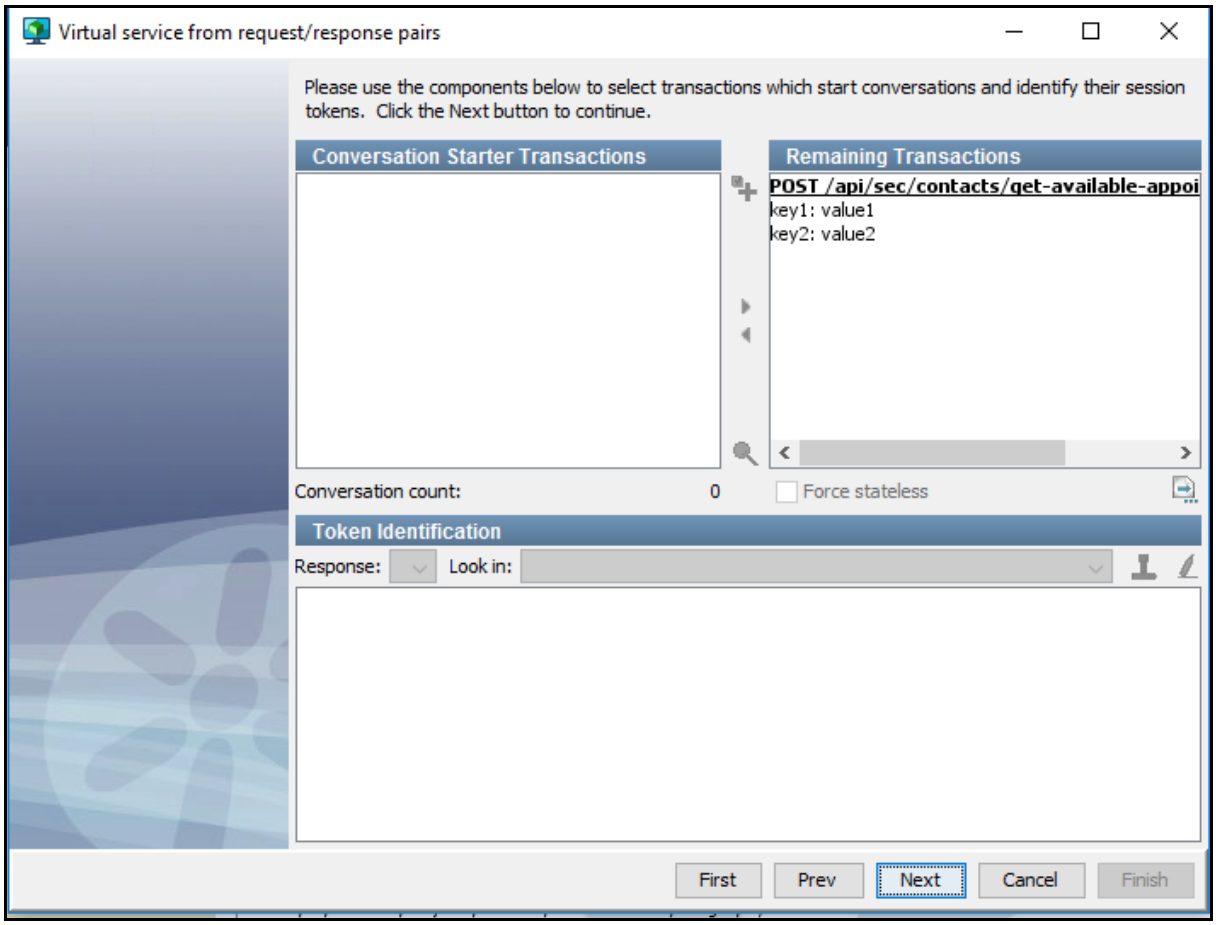
5. Appropriate data protocols for the request and response will be defaulted and click Next button



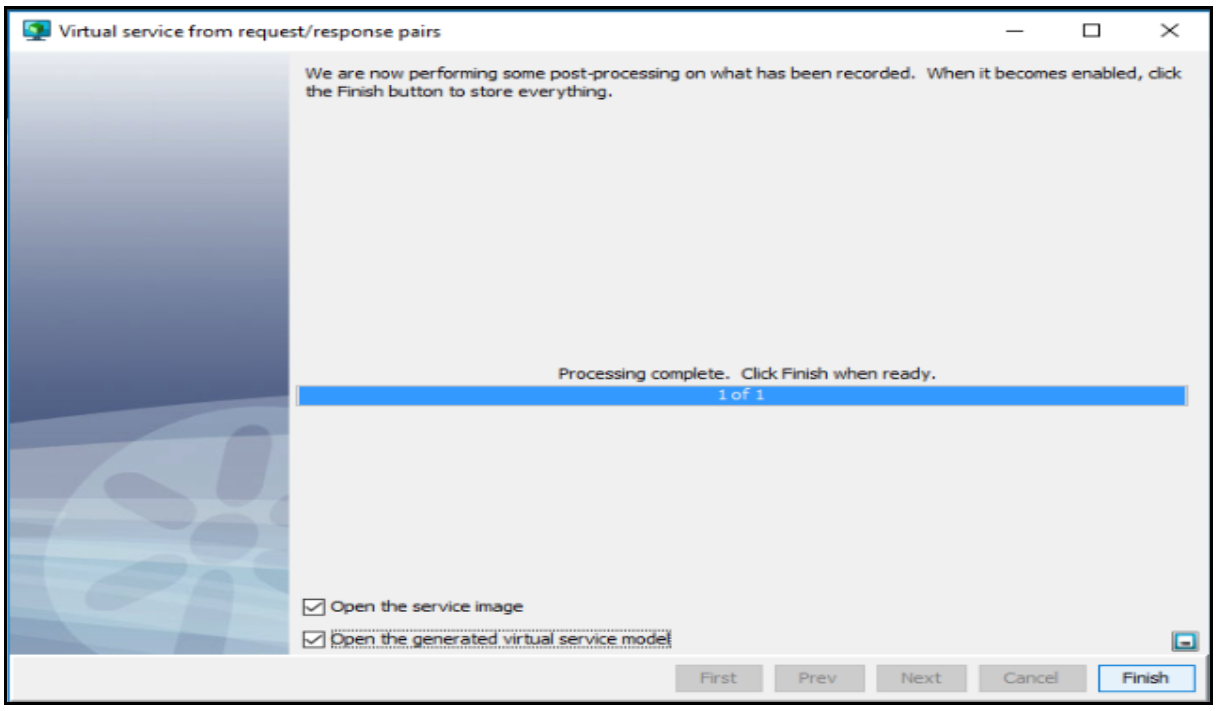
6. Appropriate URI rule will get generated. Rule can be modified by going to Properties option. Click Next button.



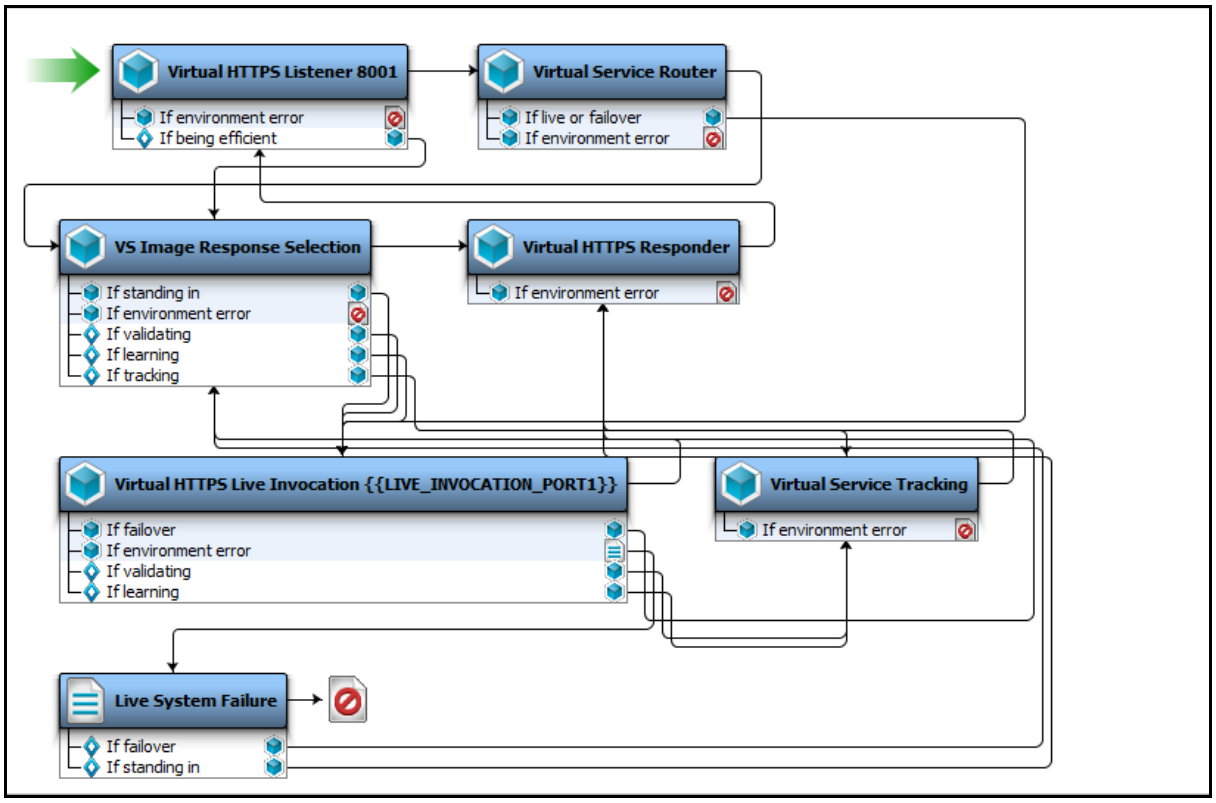
7. Click Next button



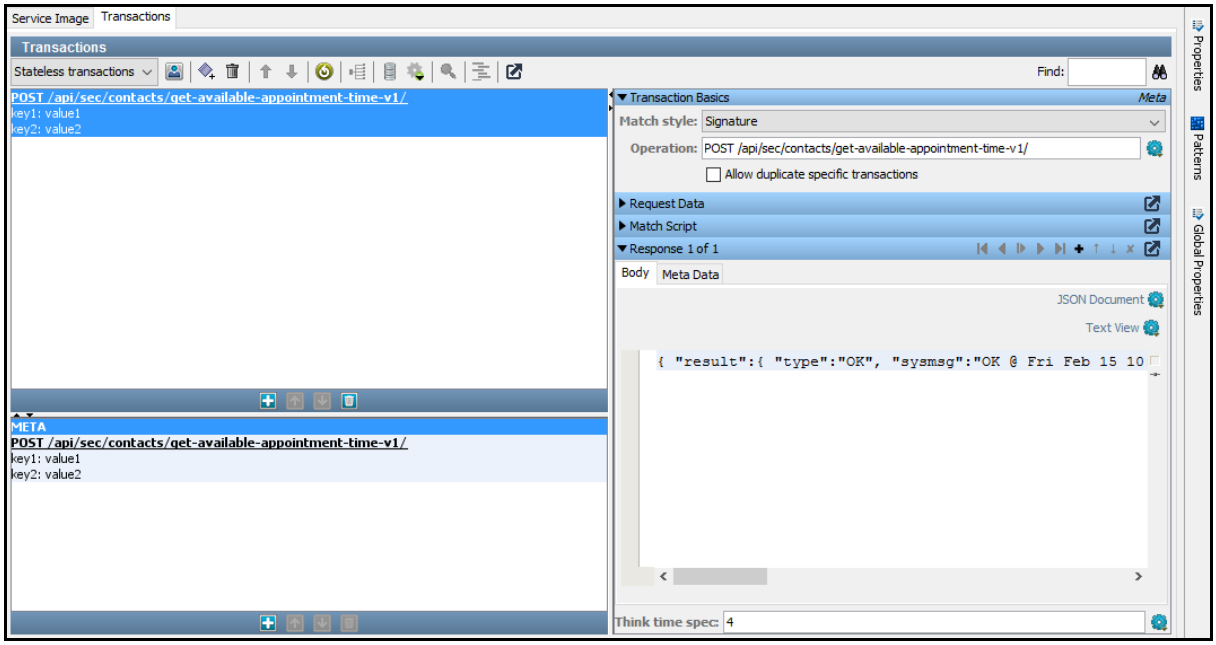
8. Select checkboxes to open VSI and VSM and click Finish button



9. VSM Created

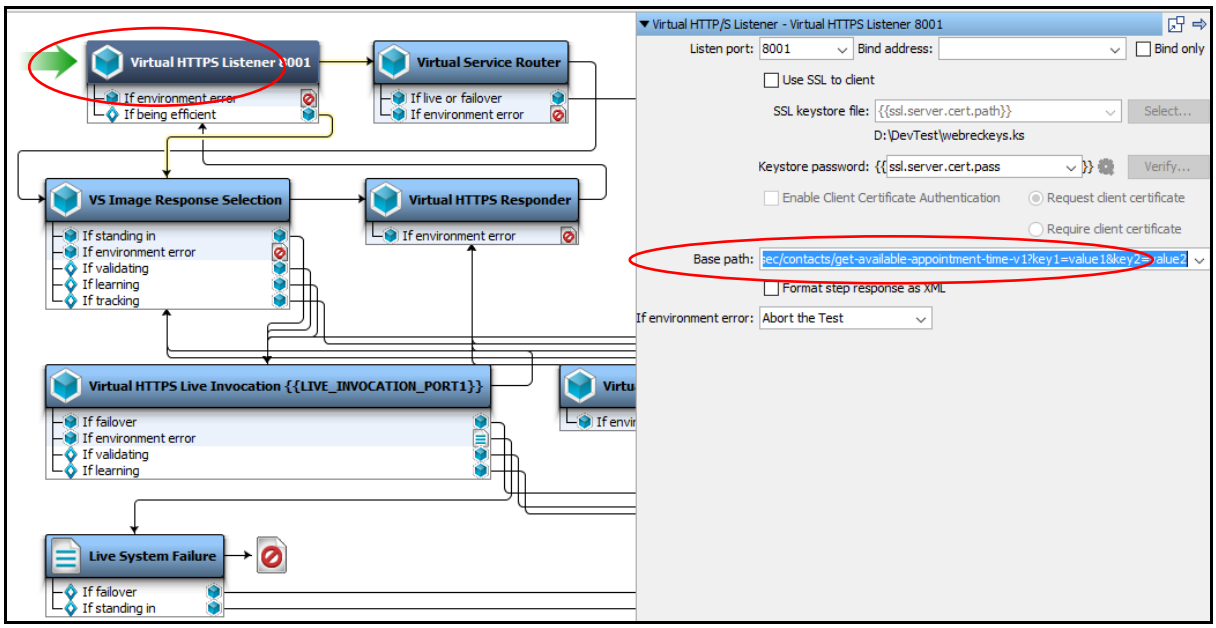


10. VSI Created

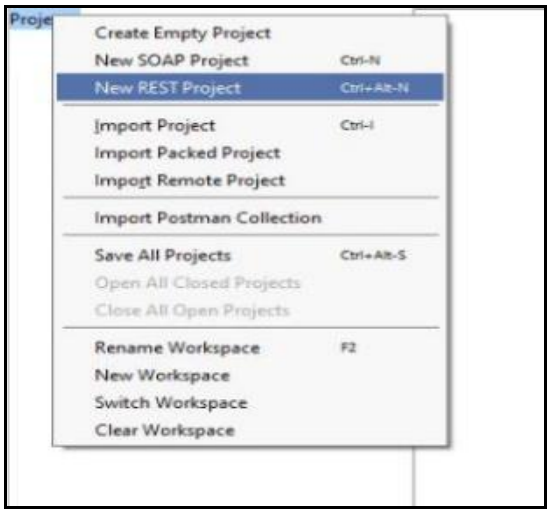


Note: Now we will hit the same request from SOAP UI and test whether the response generated is same as in Virtual Service

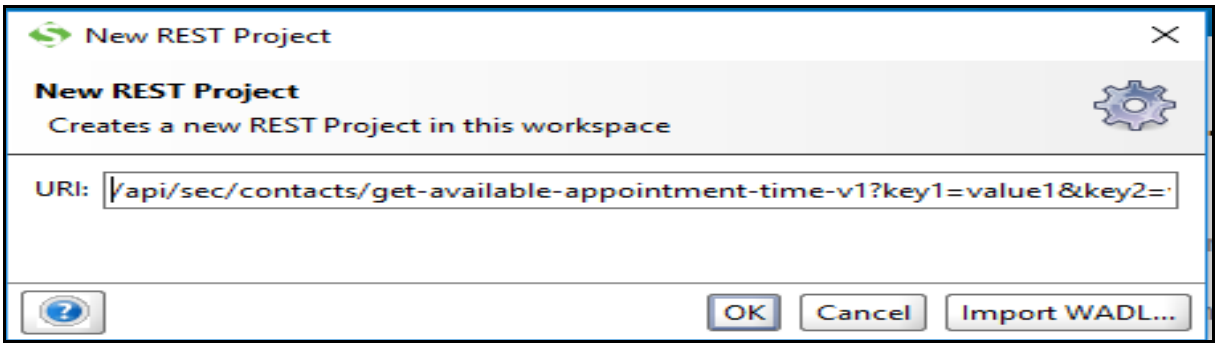
11. Double click the listener port and copy the base path.



12. Go to SOAP UI > Projects > New Rest Project

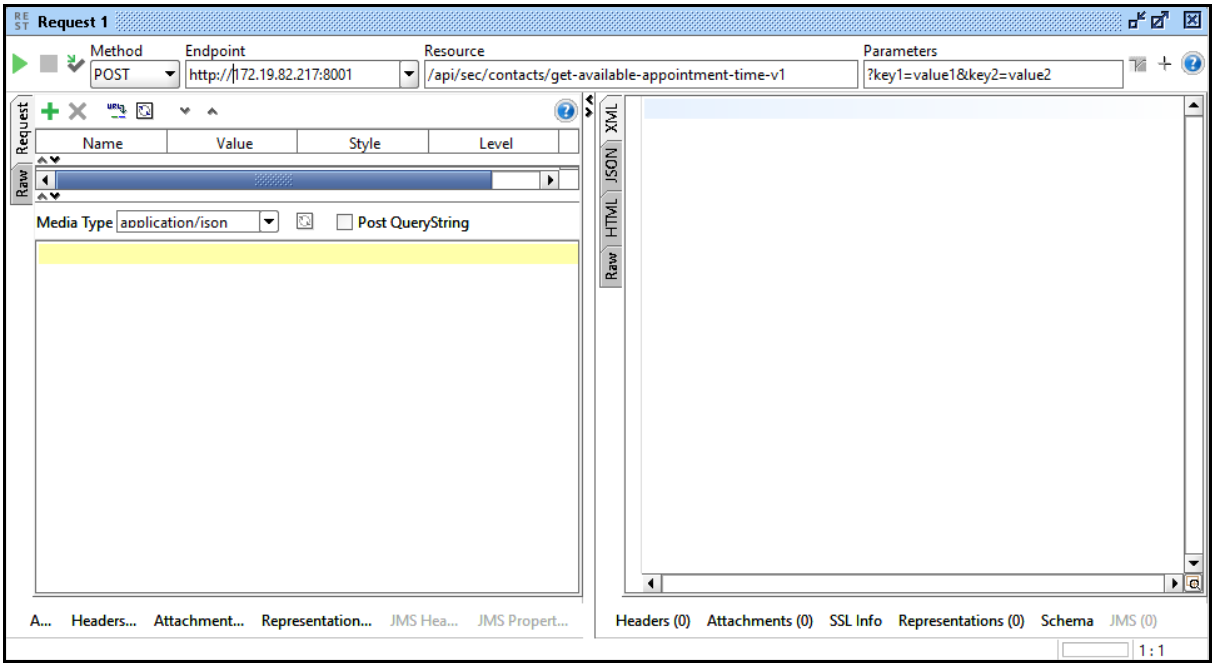


13. Enter the copied base path from VSM and click on ok.

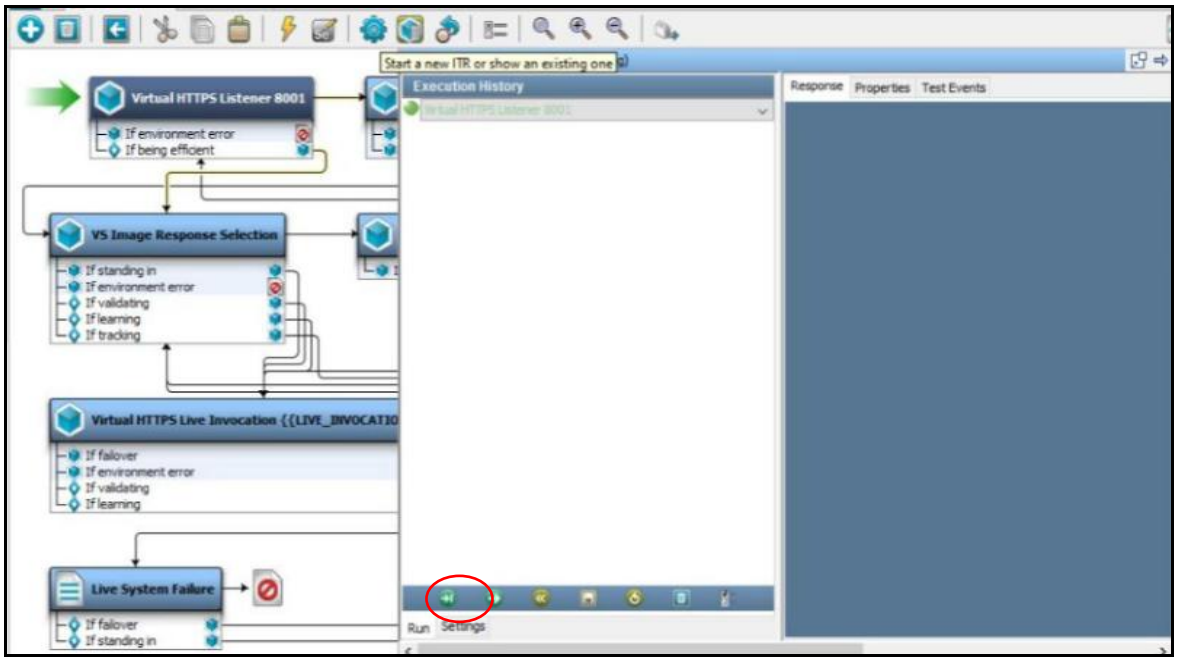


14. Enter the necessary endpoint, method, resource and parameters details

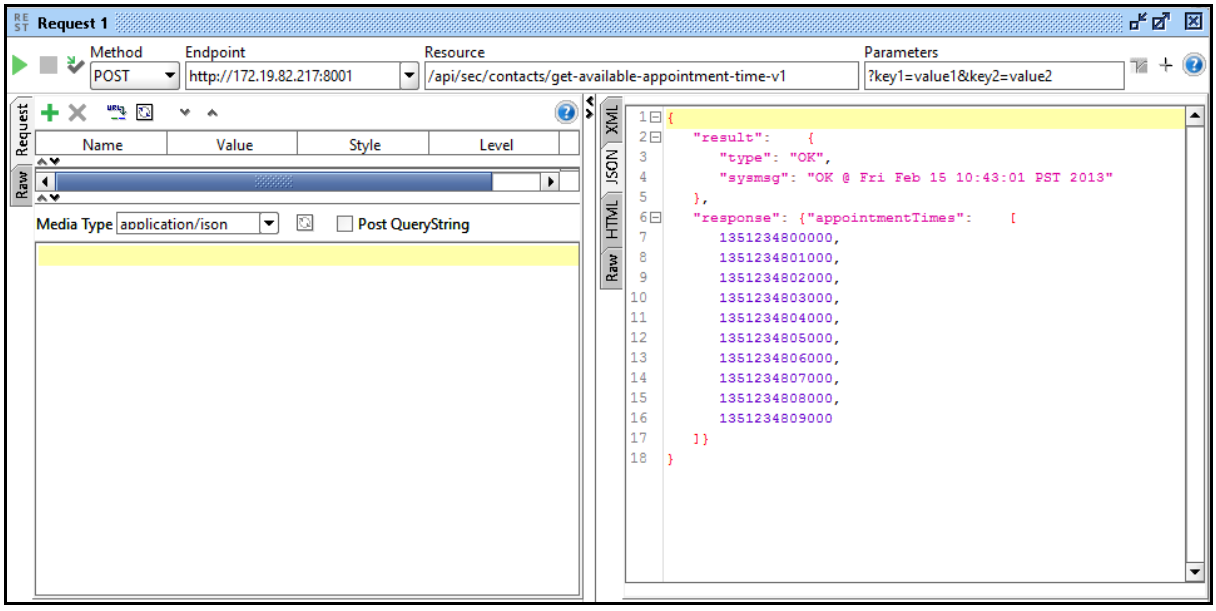
- Endpoint: Transport protocol://target host: listener port.
- Method: POST
- Resource: Base Path
- Parameters



15. Go to VSM and start ITR and click on automatically execute the test



16. Click Run button. SOAP UI will hit the same response as in Virtual Service

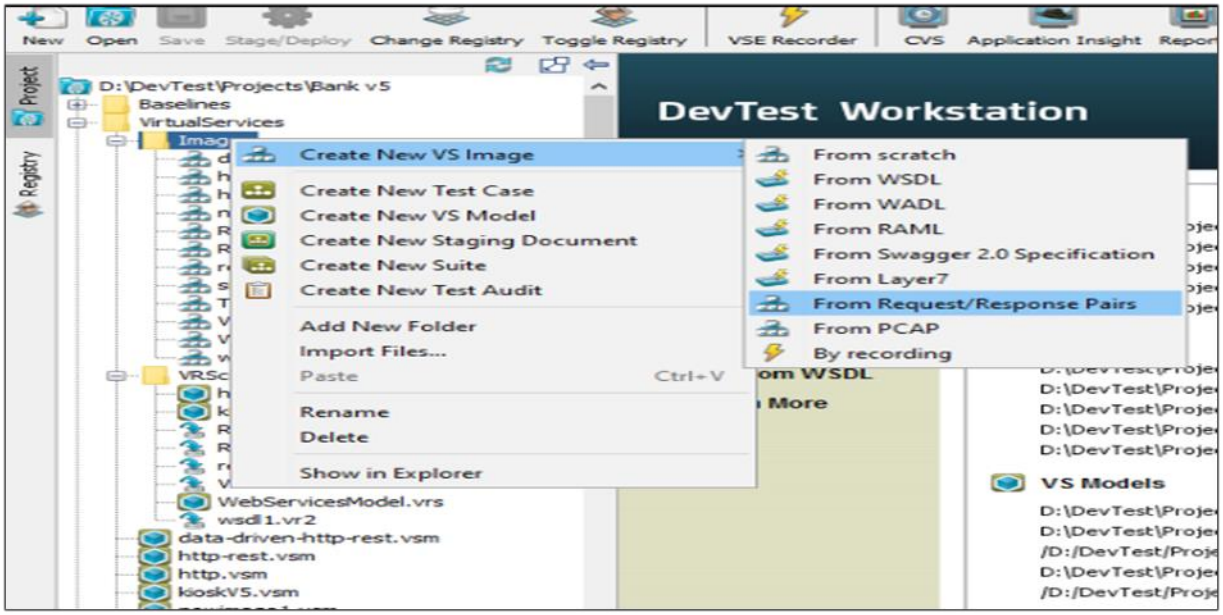


Request and Response Pair files

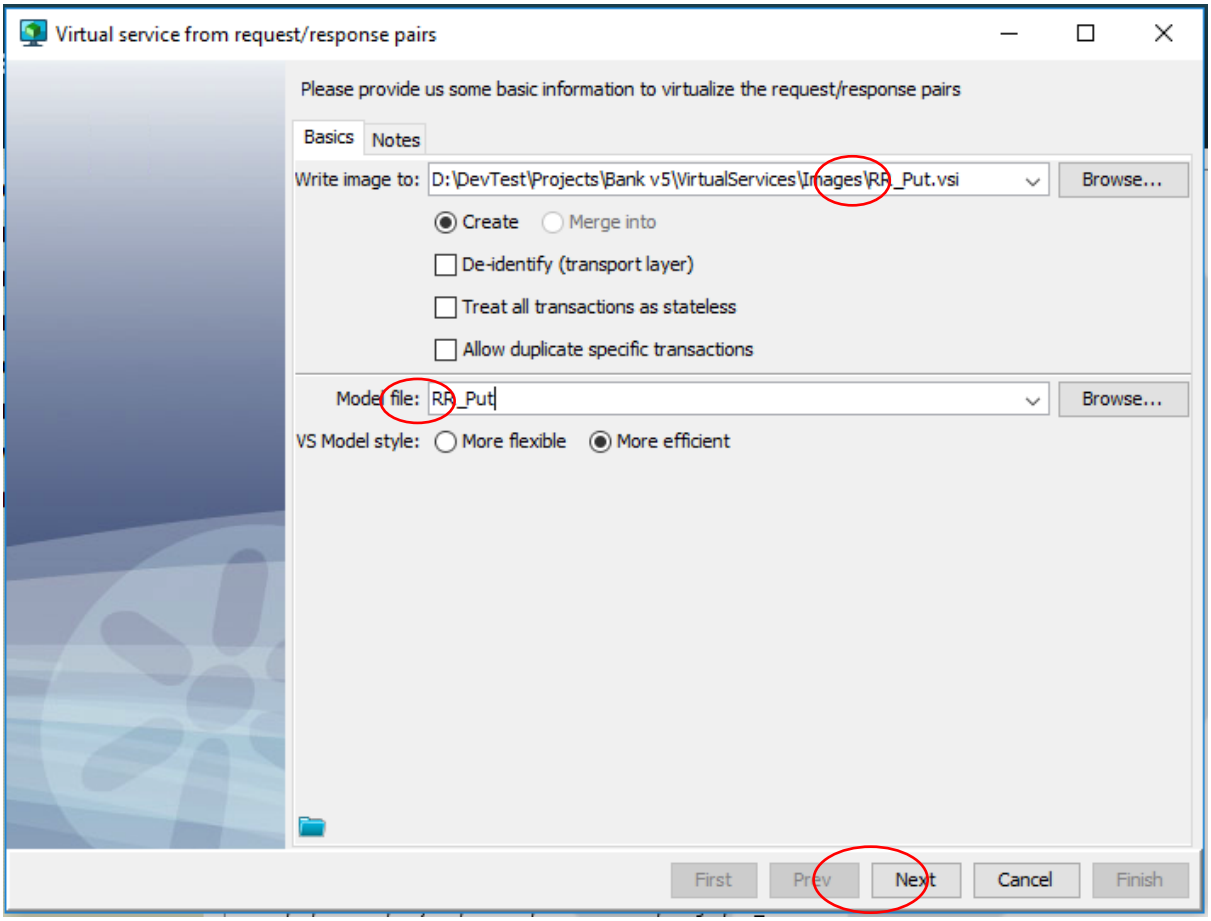
14.3 Request and Response Pair Using Rest Put Method

PUT is most-often utilized for update capabilities, PUT to a known resource URI with the request body containing the newly-updated representation of the original resource.

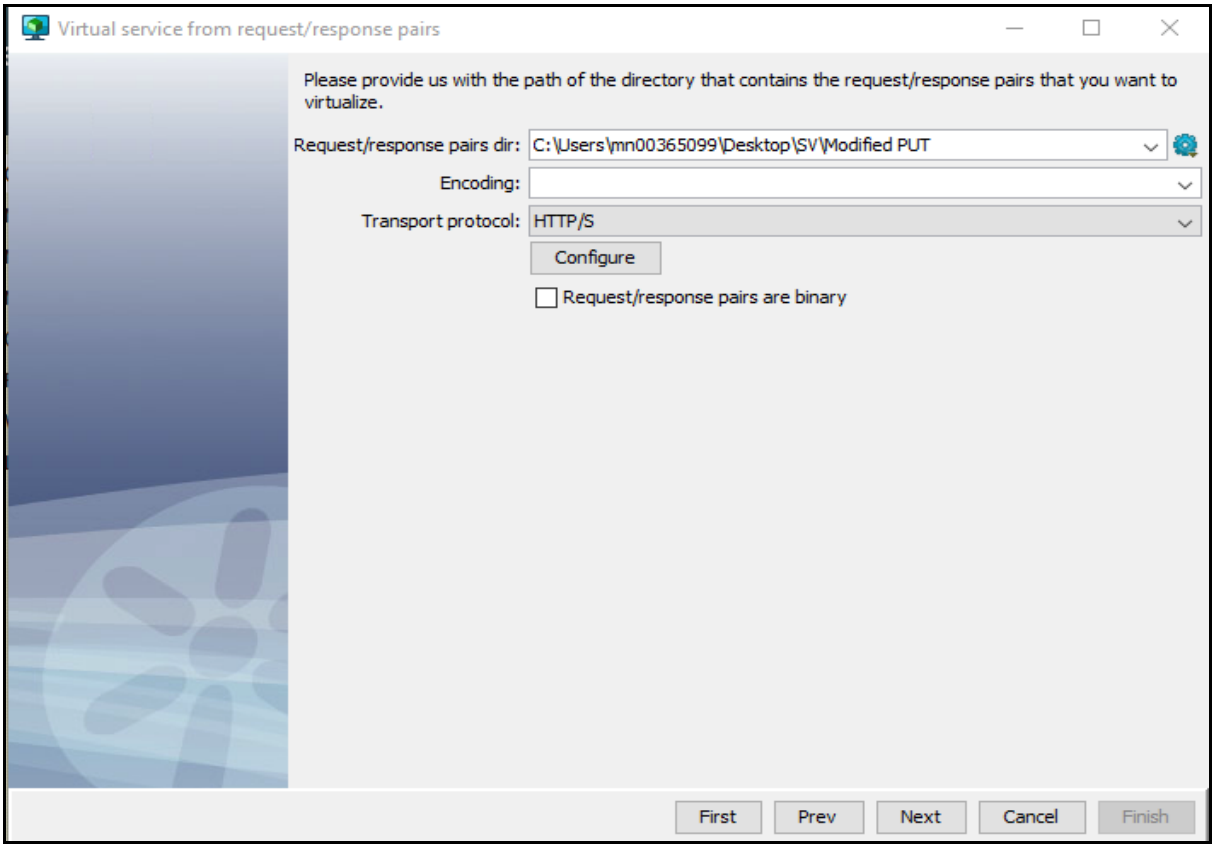
1. Select Projects > Images > Create New VS Image > From Request/Response Pairs



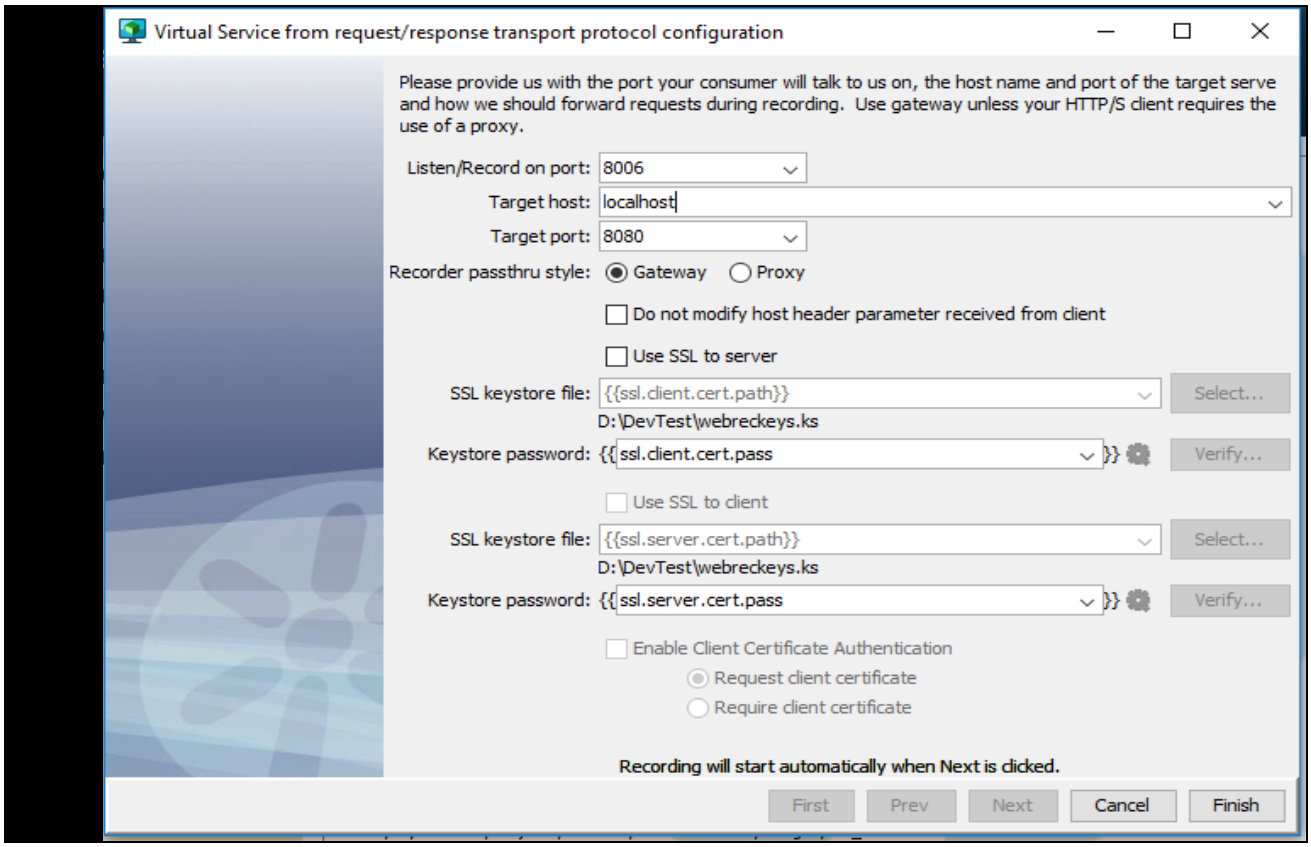
2. Enter Service Image name and Model name, click Next button



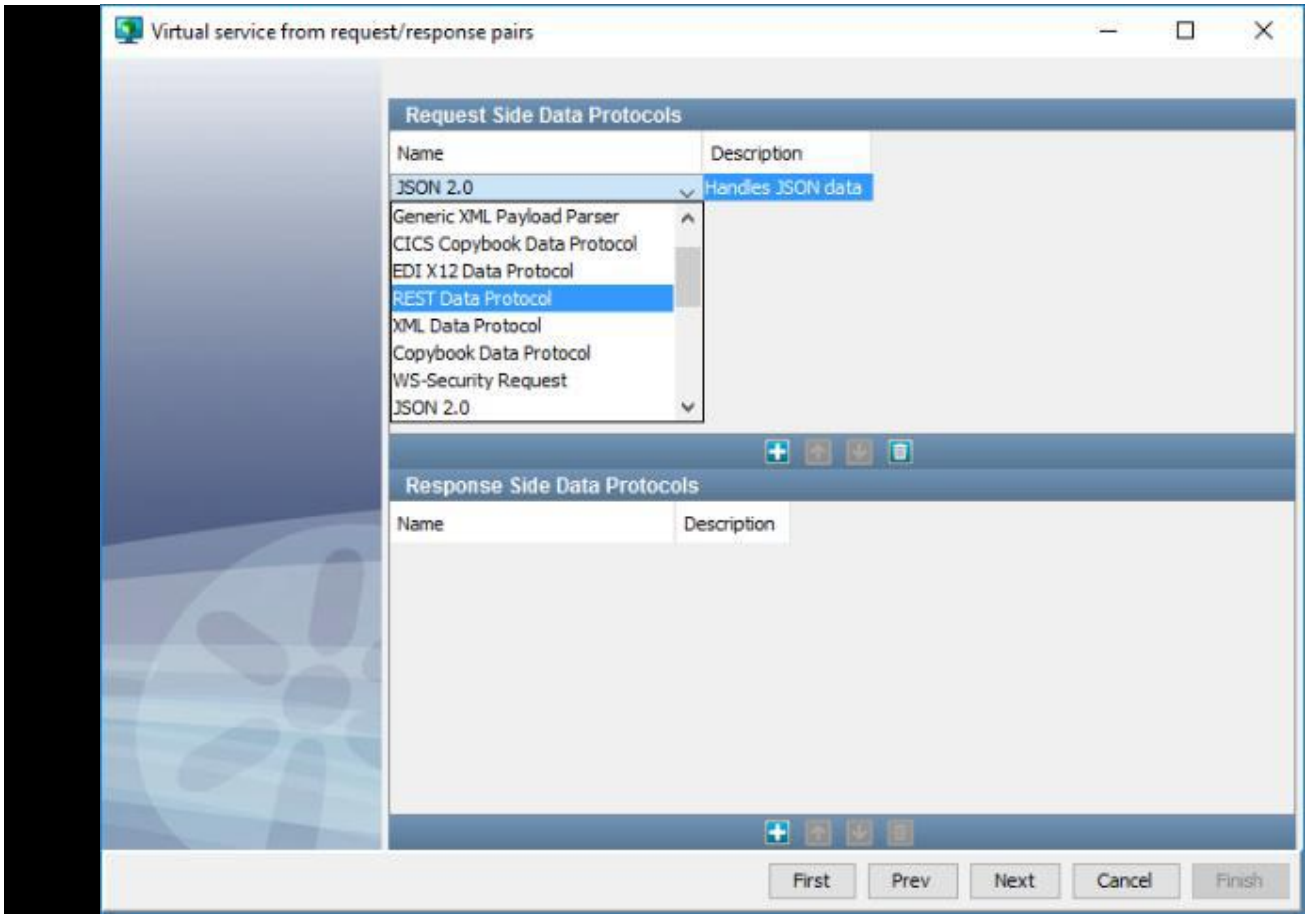
3. Browse the path for request/response pairs & select Transport protocol from the drop down list. Click Configure button



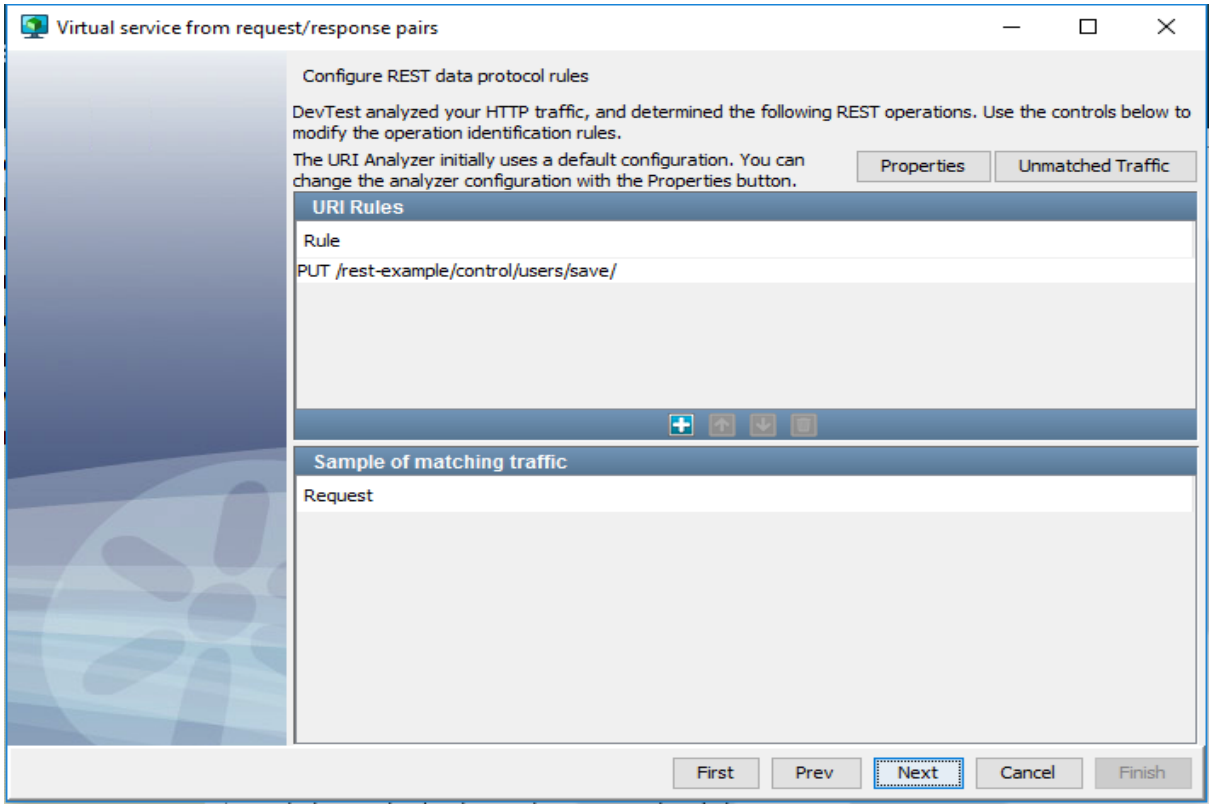
4. Provide details for Listen port, Target host & Target port and click Finish button



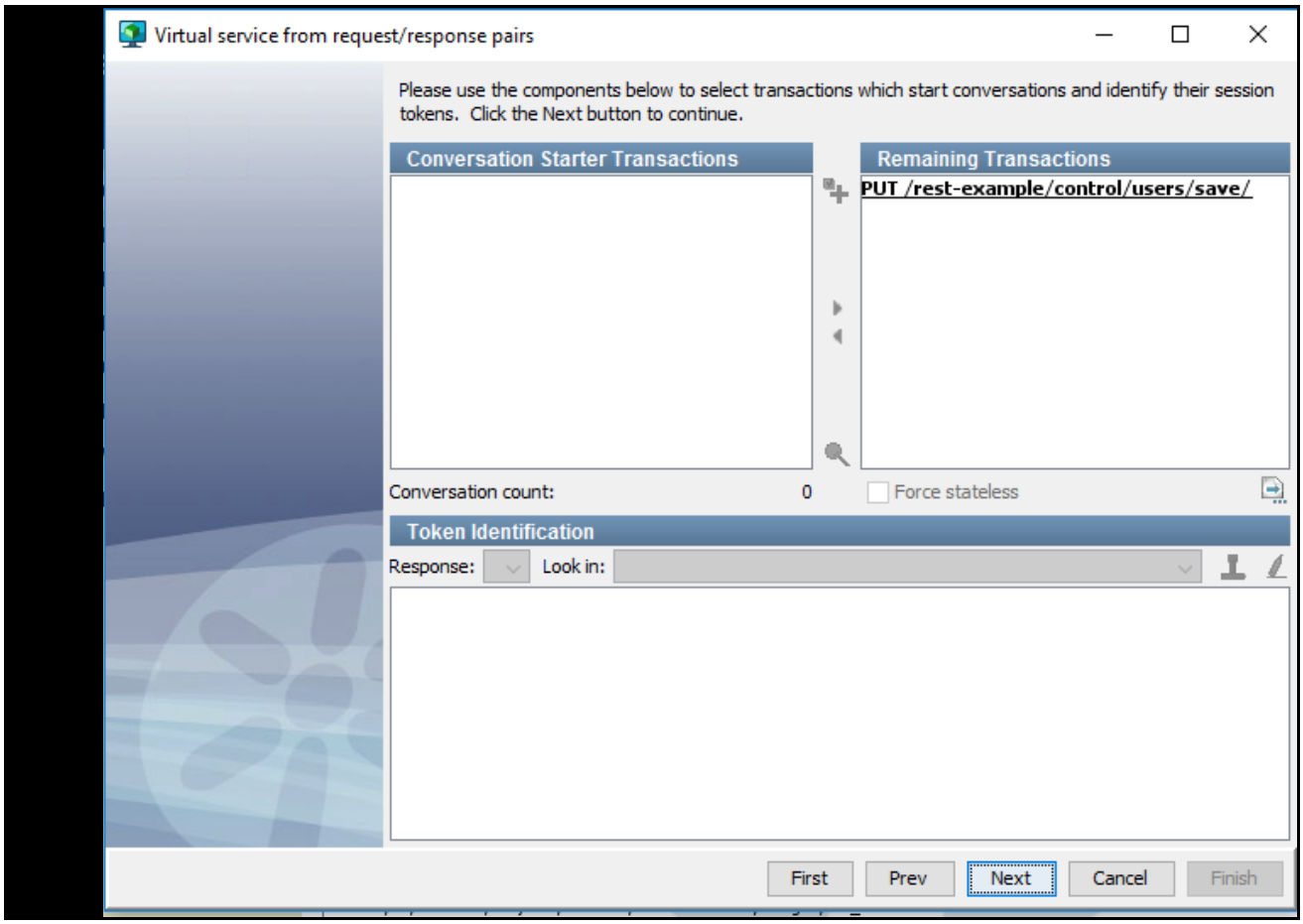
5. Select Rest Data Protocol instead of JSON protocol and click Next button



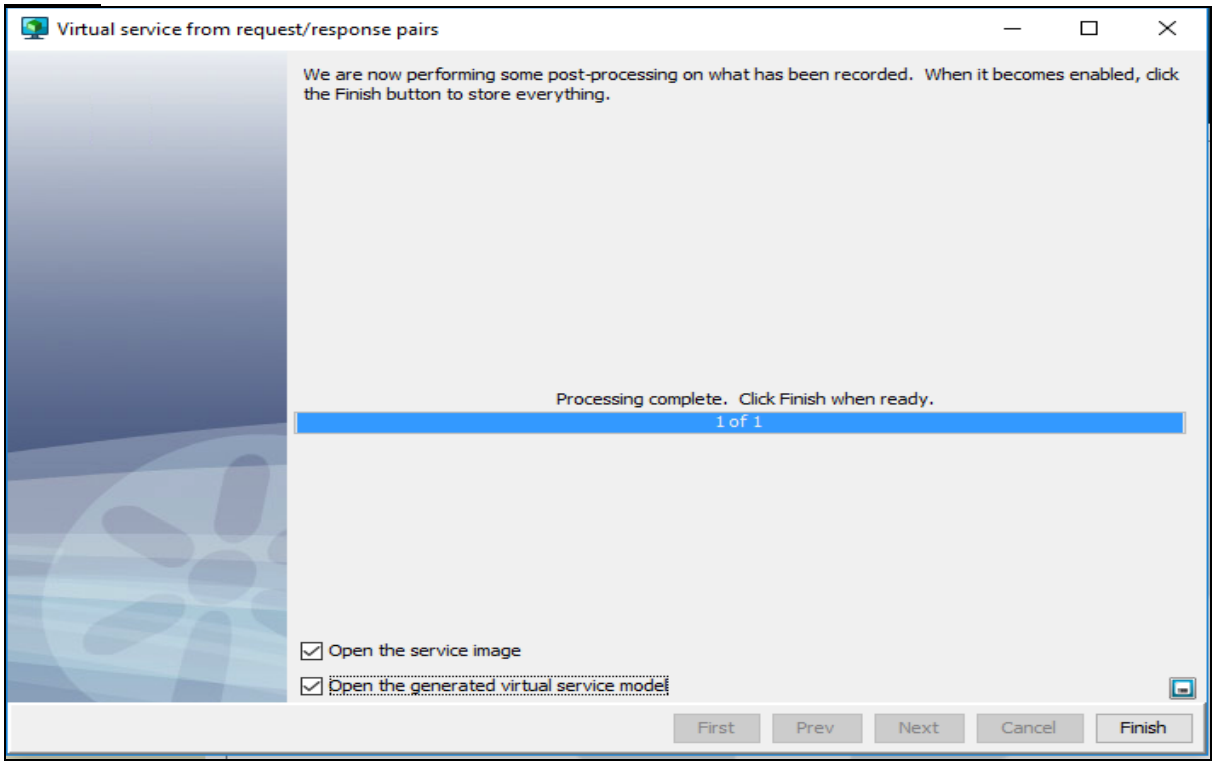
6. Appropriate URI rule will get generated. Rule can be modified by going to Properties option. Click Next button.



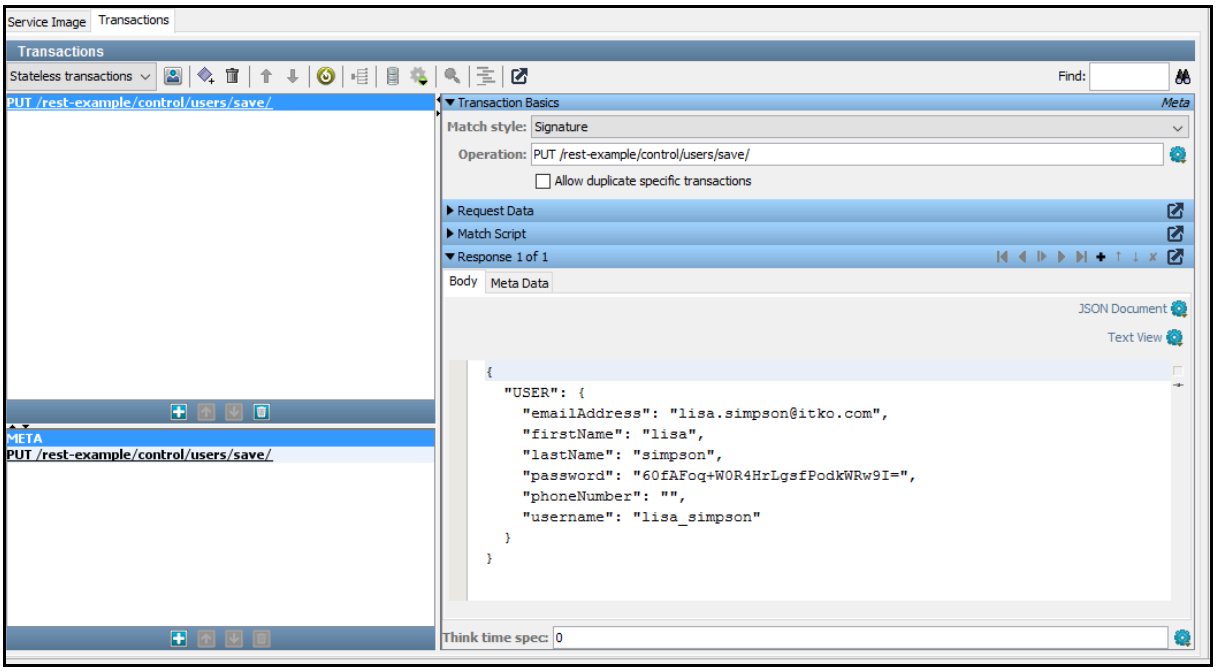
7. Click Next button



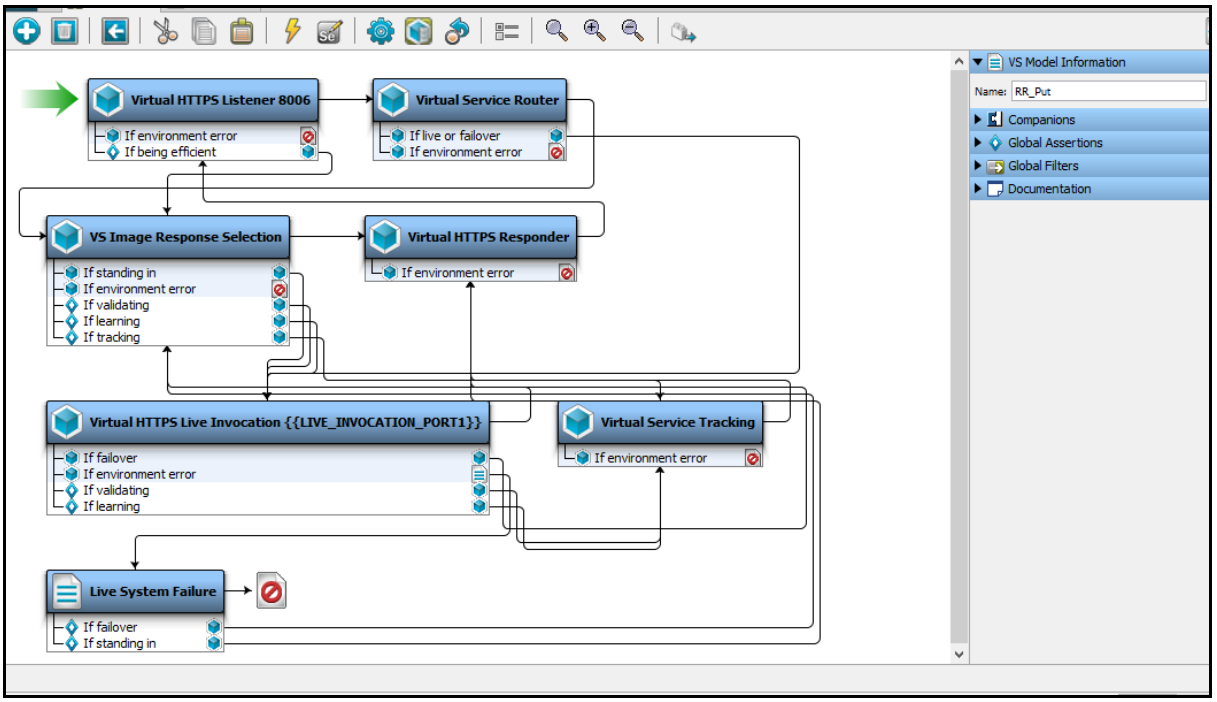
8. Select checkboxes to open VSI and VSM and click Finish button



9. Virtual Service Image created.

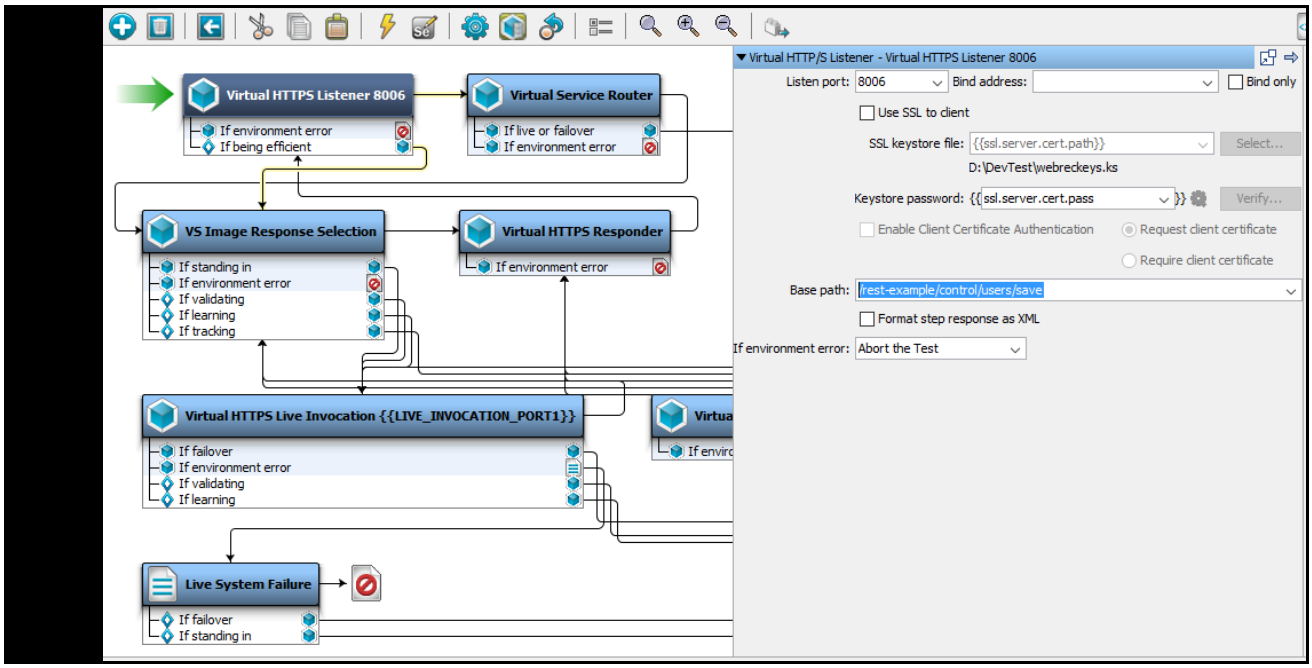


10. Virtual Service Model created

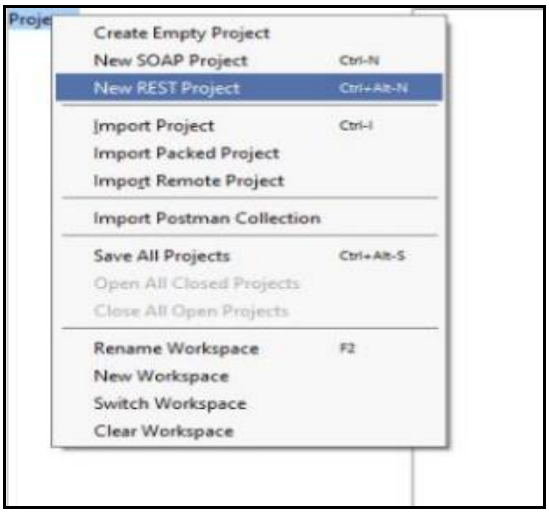


Note: Now we will hit the same request from SOAP UI and test whether the response generated is same as in Virtual Service.

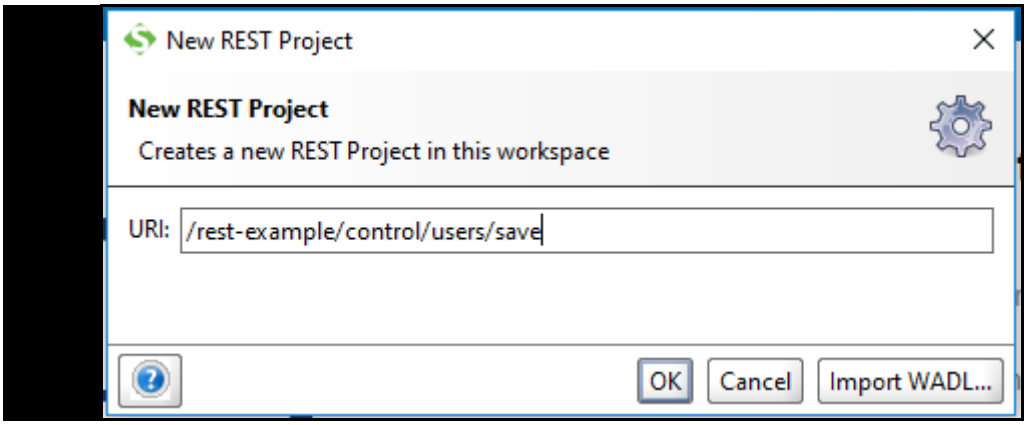
11. Double click the listener port and copy the base path.



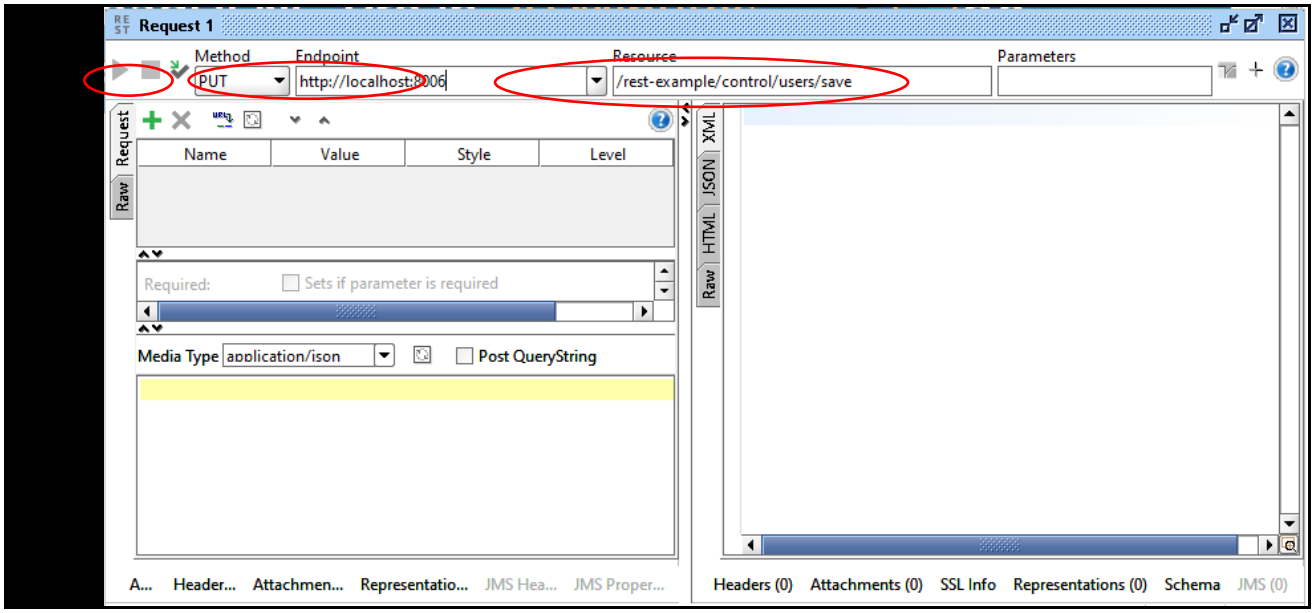
12. Go to SOAP UI > Projects > New Rest Project



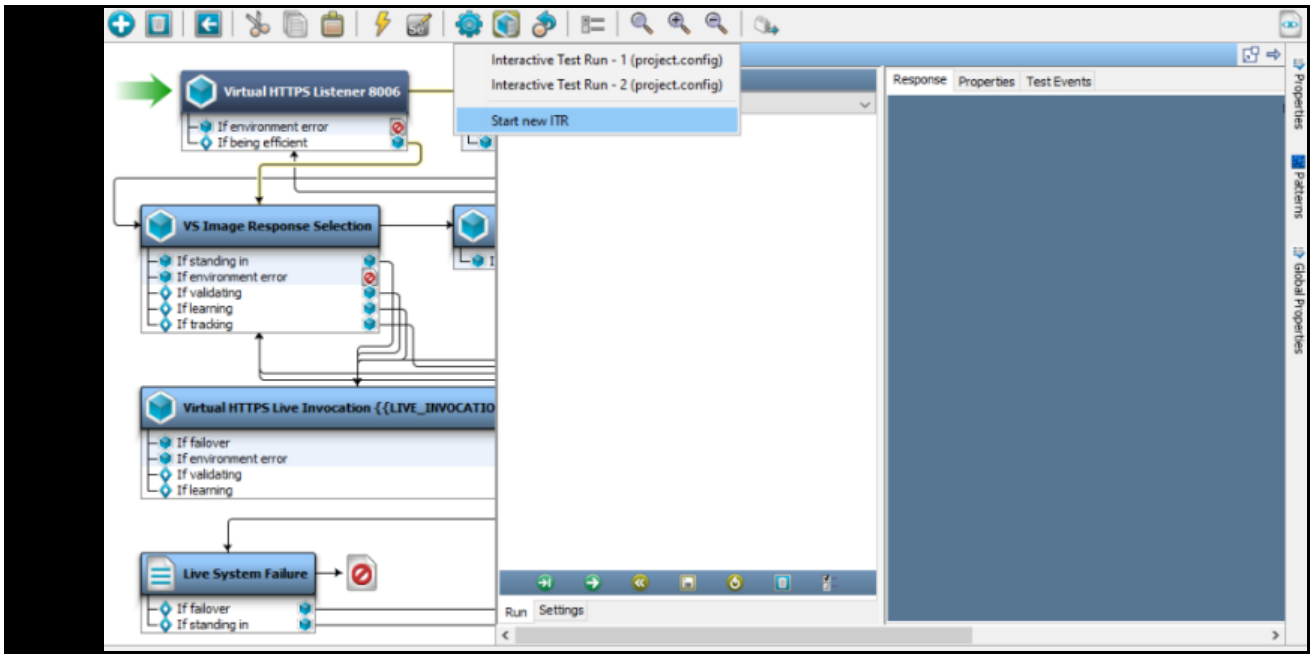
13. Enter the copied base path from VSM and click on ok



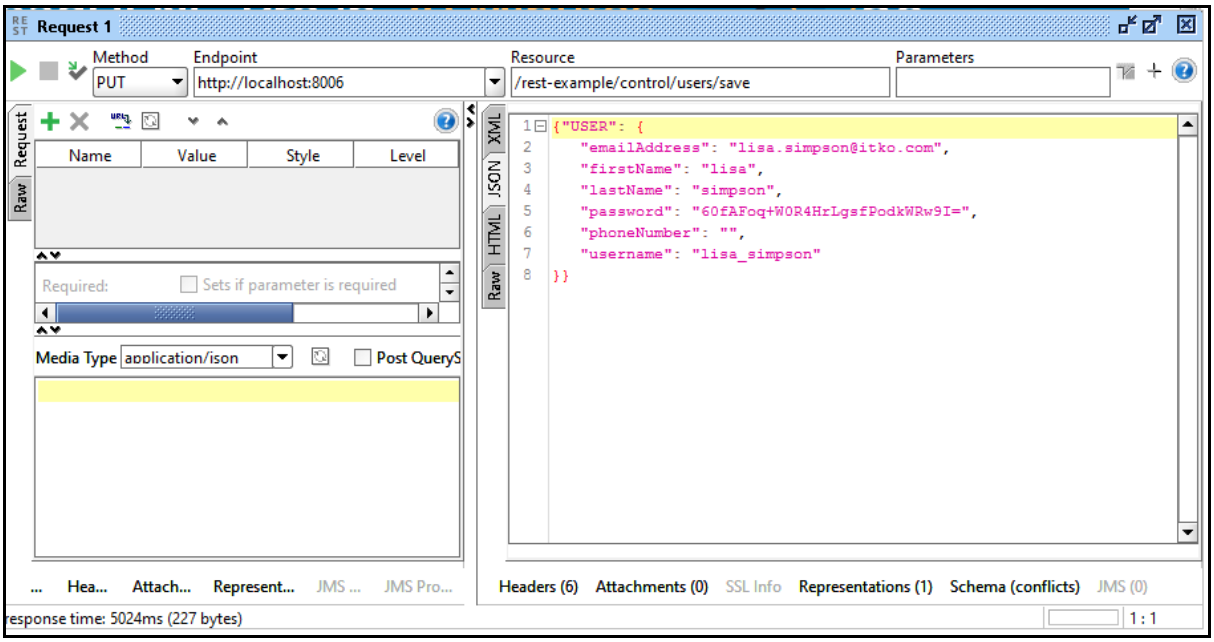
14. Enter the necessary endpoint, method, resource and parameters details
    - Endpoint: Transport protocol://target host: listener port.
    - Method: PUT
    - Resource: Base Path



15. Go to VSM and start ITR and click on automatically execute the test.



16. Click on run button in SOAP UI

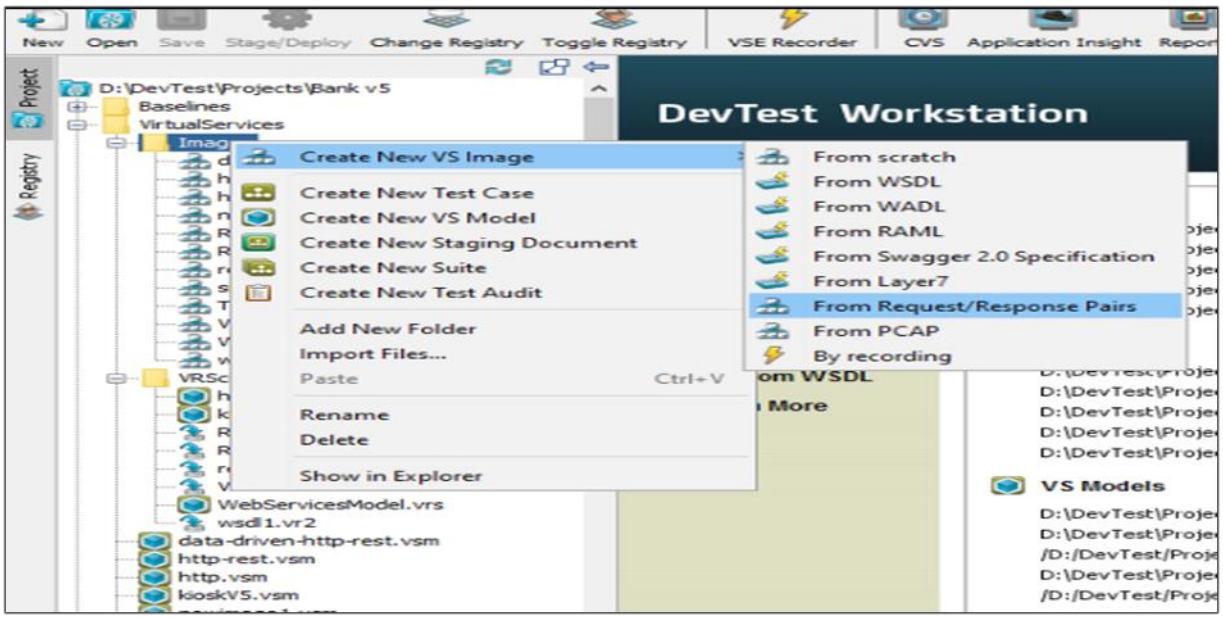


Sample: Request and Response Pair files

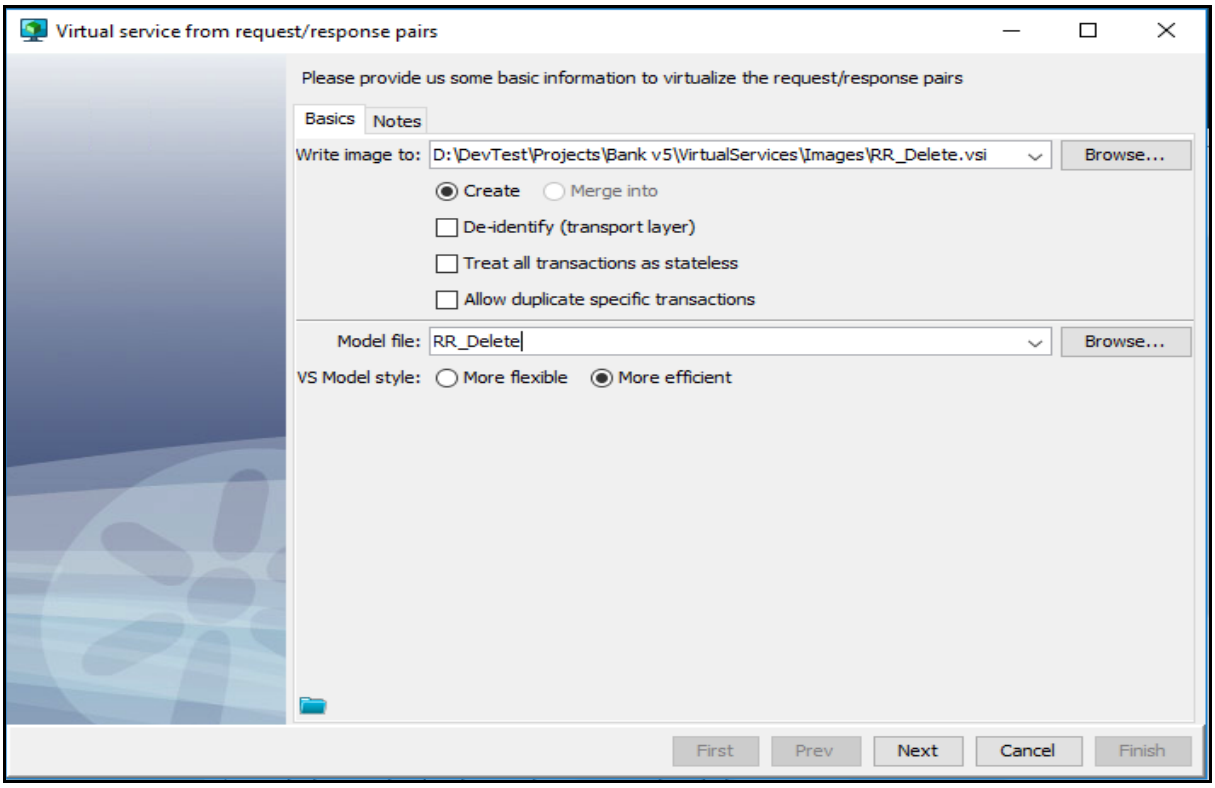
14.4 Request and Response Pair Using Rest Delete Method

DELETE method is used to delete a resource identified by a URI.

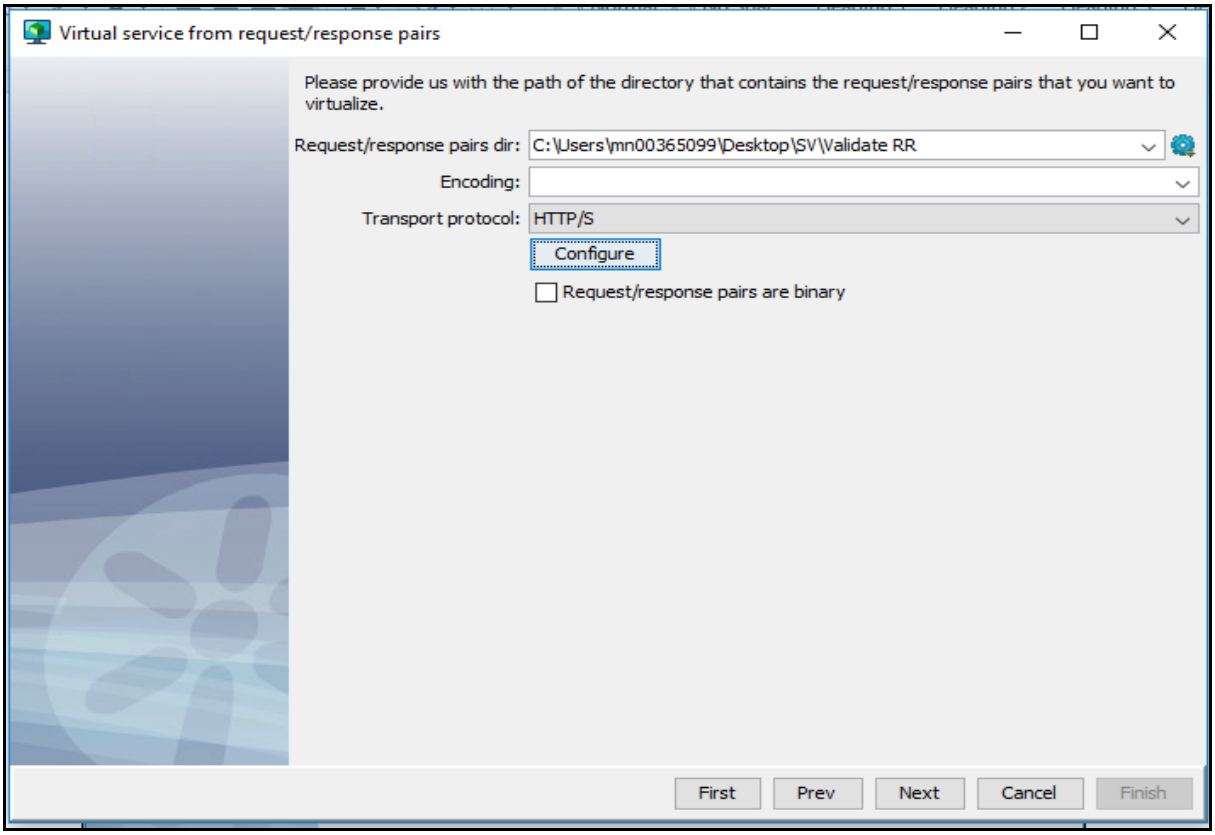
1. Select Projects > Images > Create New VS Image > From Request/Response Pairs



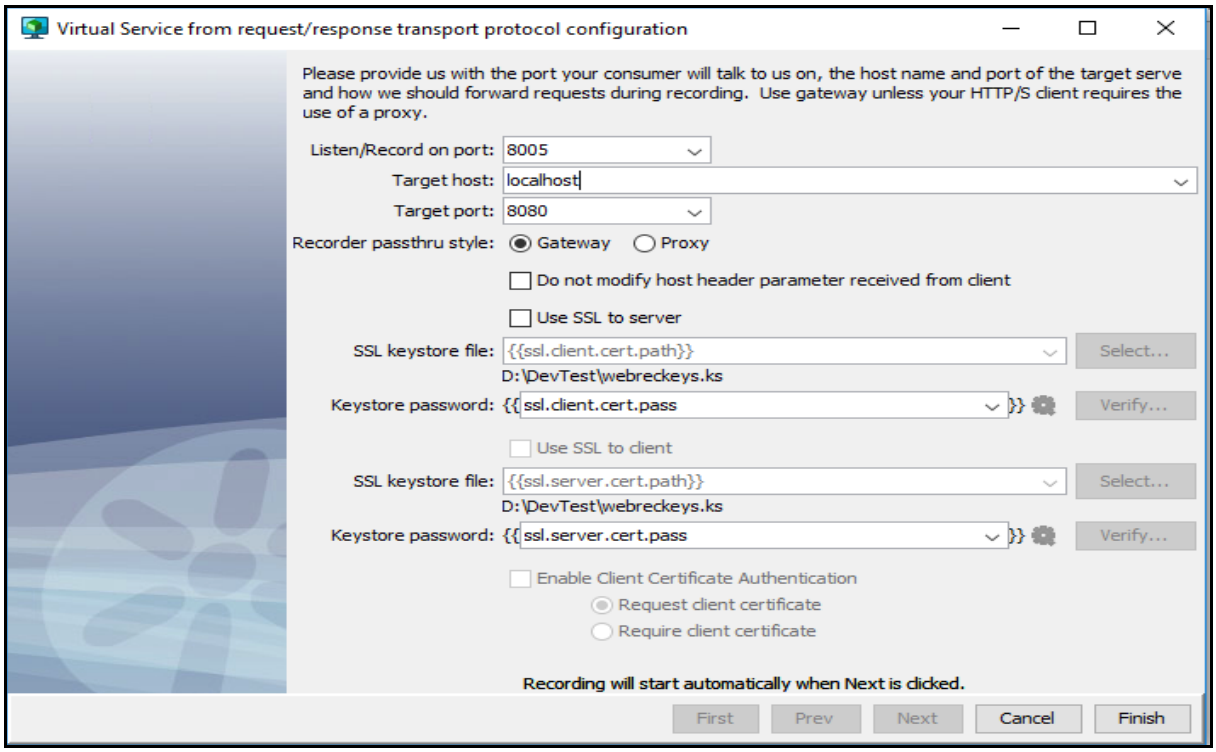
2. Enter Service Image name and Model name, click Next button



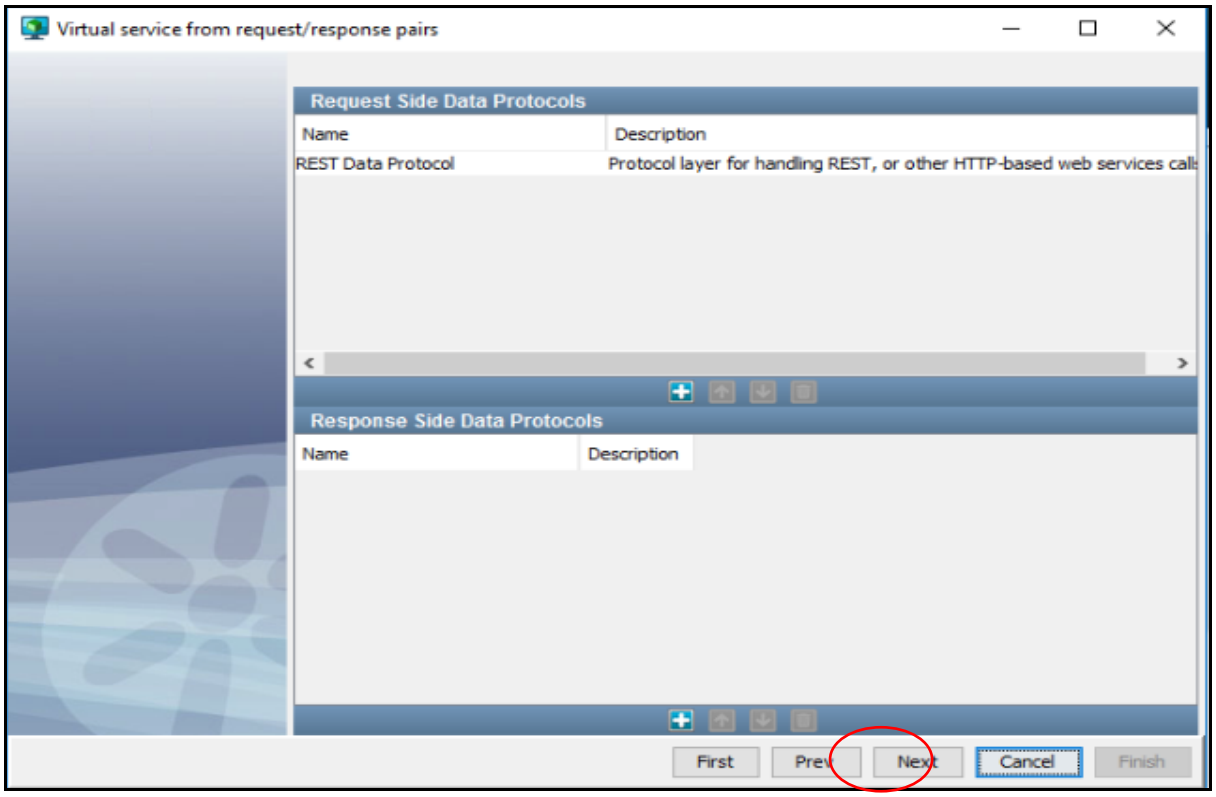
3. Browse the path for request/response pairs & select Transport protocol from the drop down list. Click Configure button



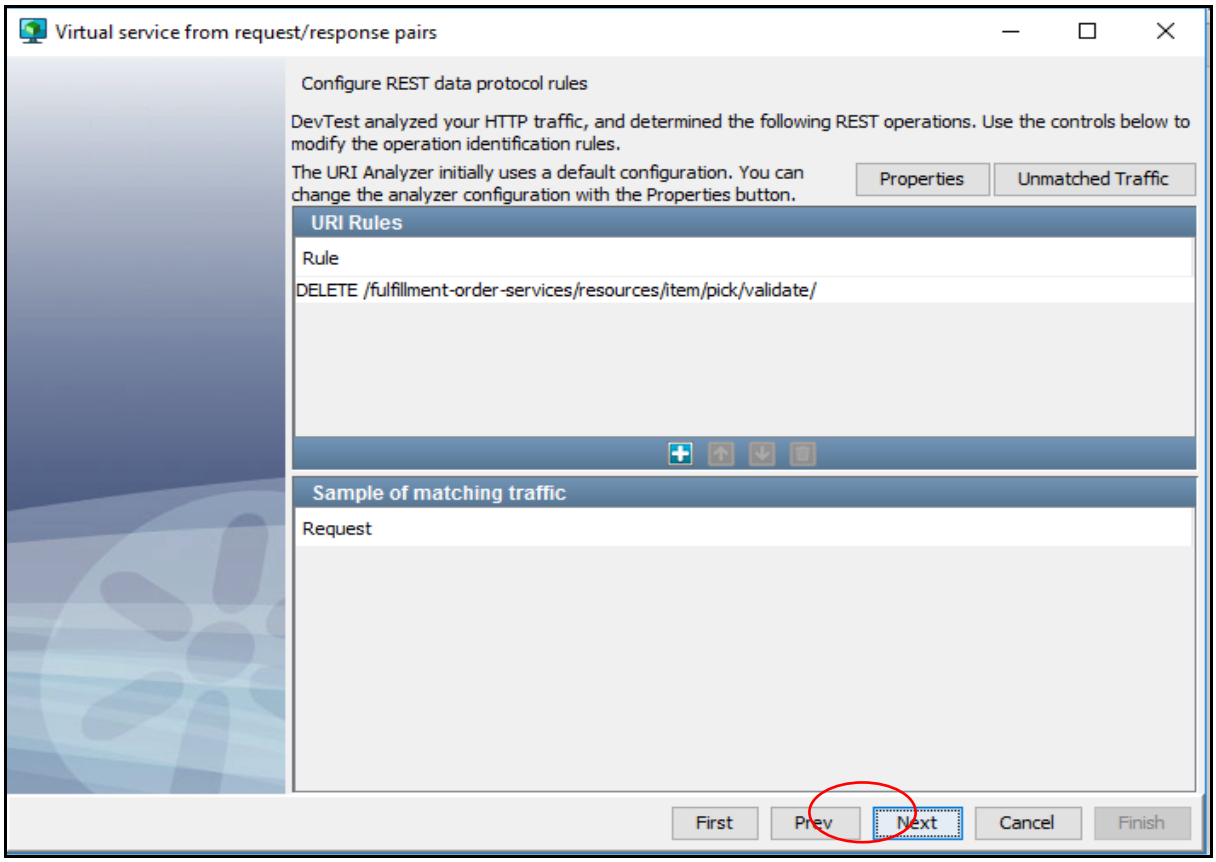
4. Provide details for Listen port, Target host & Target port and click Finish button



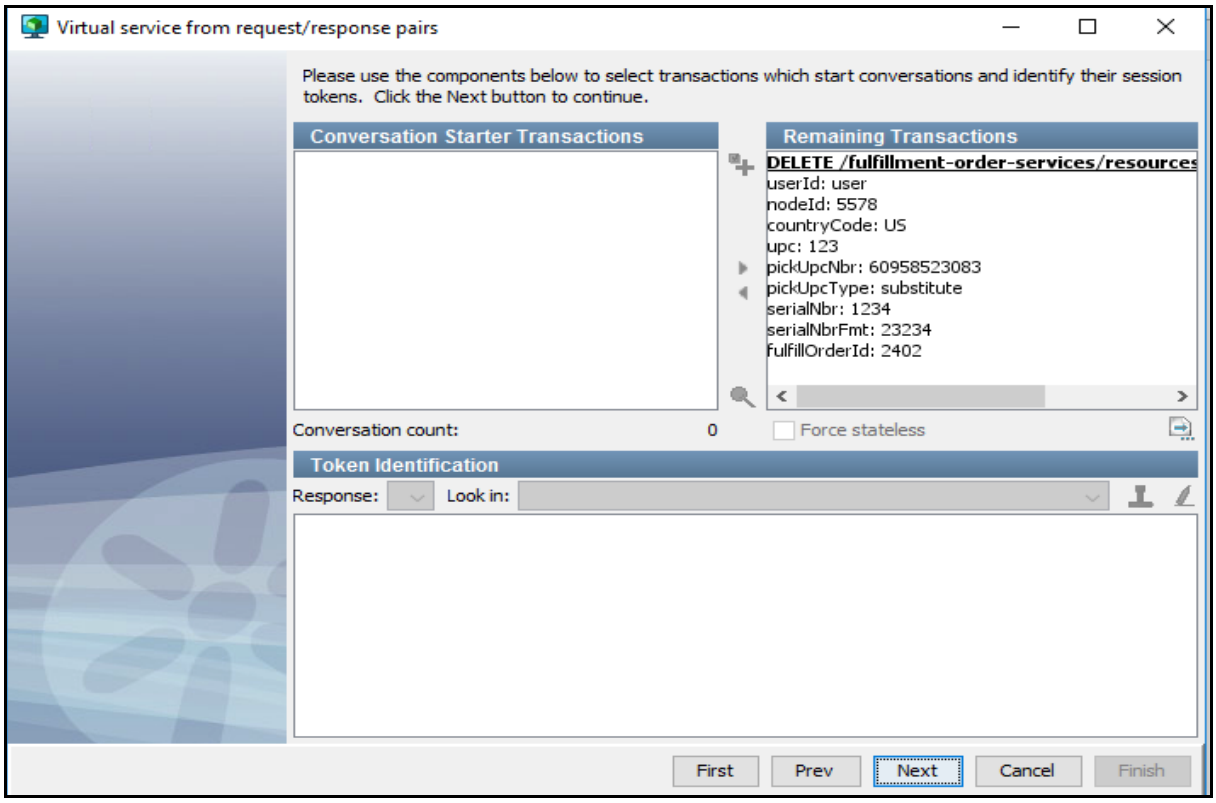
5. Appropriate data protocols for the request and response will be defaulted and click Next button



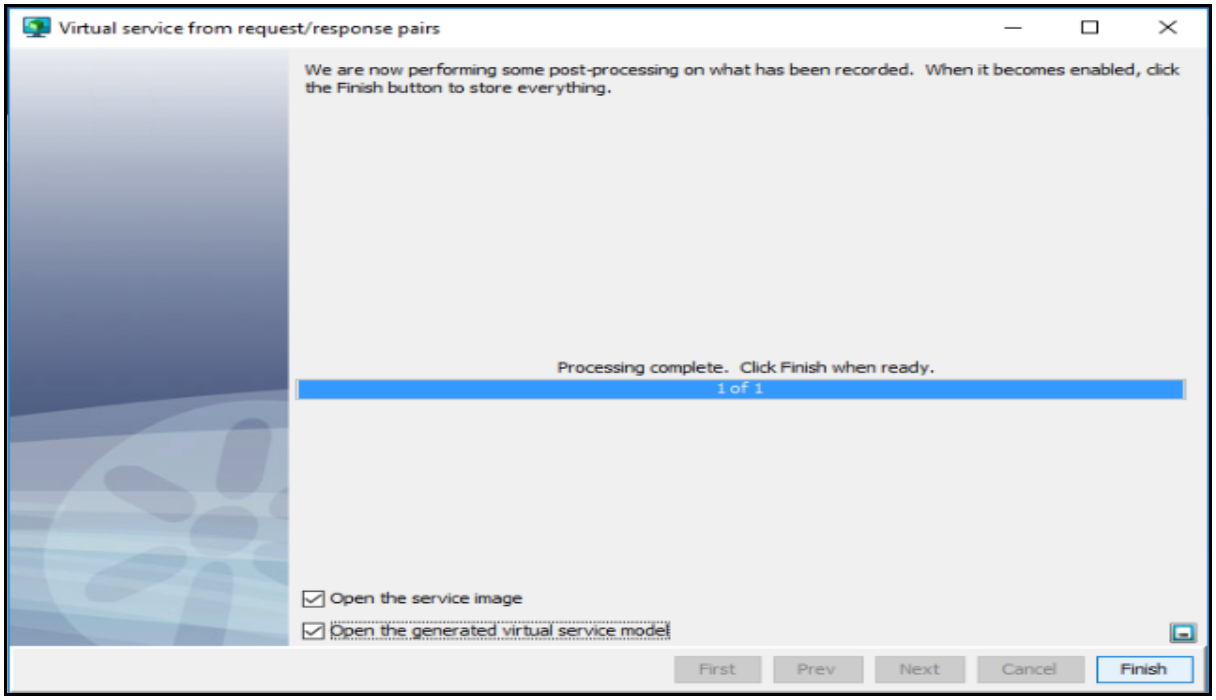
6. Appropriate URI rule will get generated. Rule can be modified by going to Properties option. Click Next button.



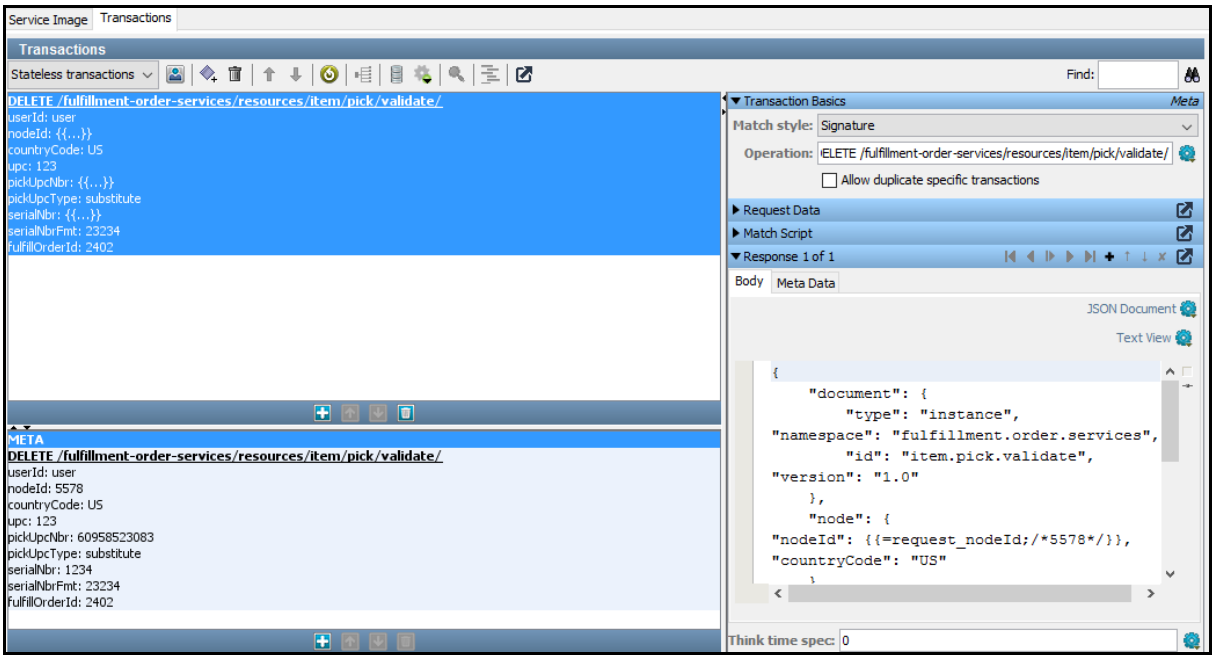
7. Click Next button



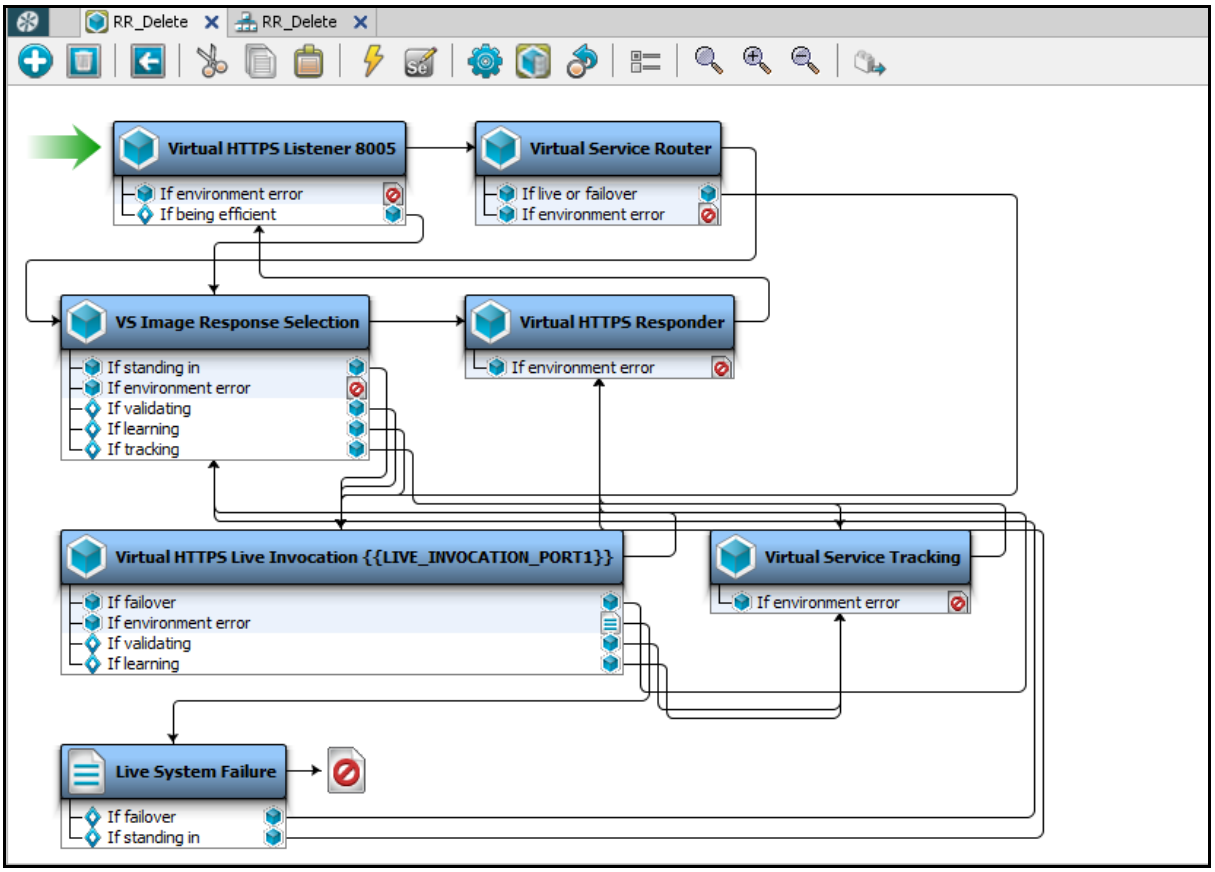
8. Check open VSI and VSM checkboxes and click Finish button



9. Virtual Service Image created

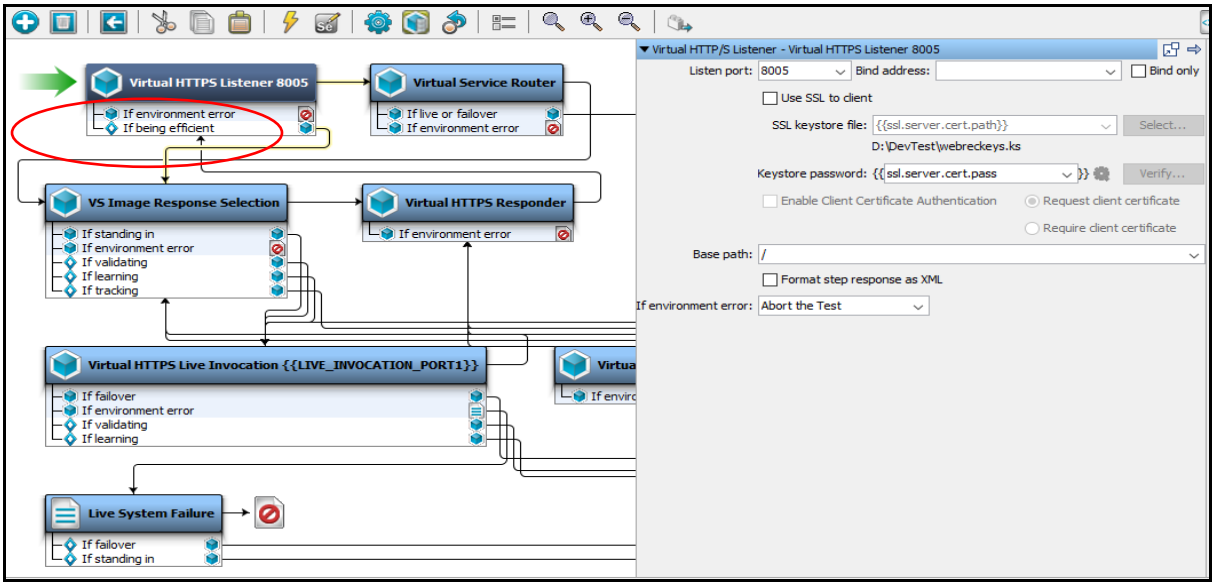


10. Virtual Service Model created

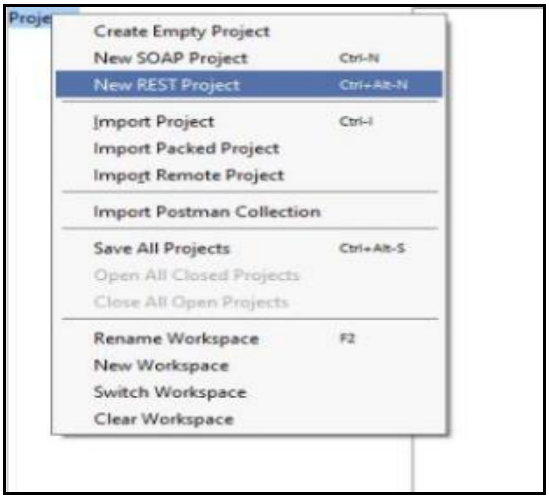


Note: Now we will hit the same request from SOAP UI and test whether the response generated is same as in Virtual Service.

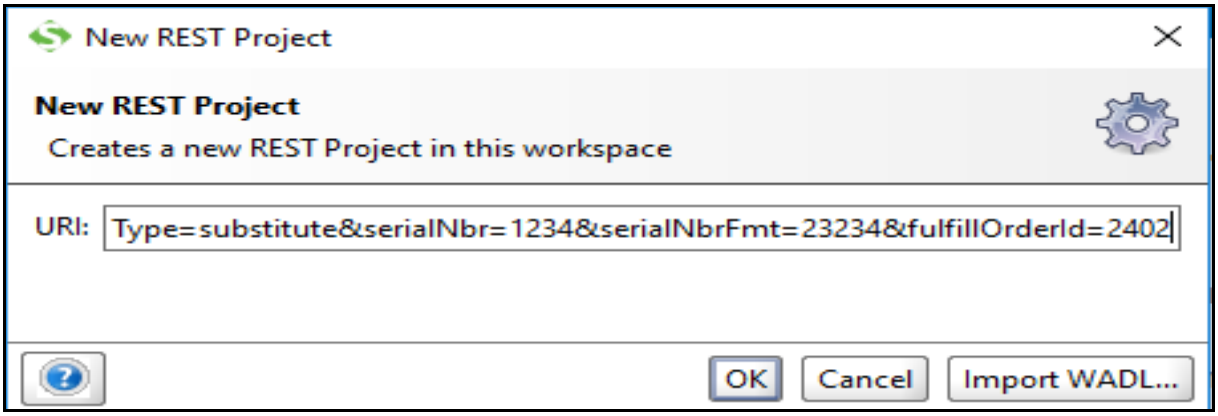
11. Double click the listener port and copy the base path



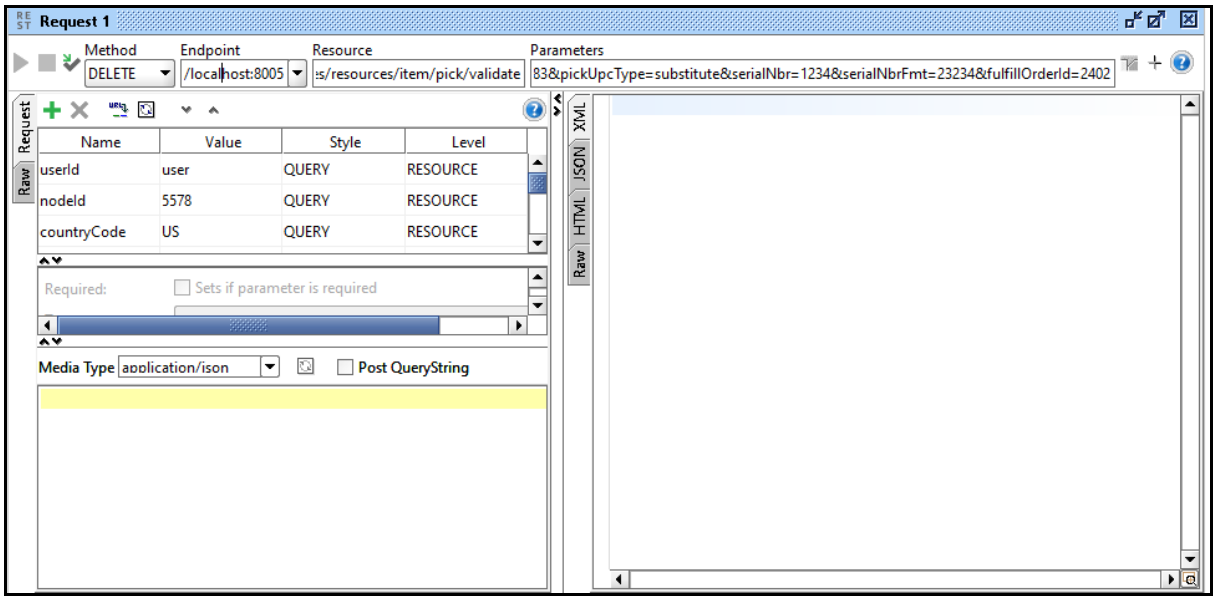
12. Go to SOAP UI > Projects > New Rest Project



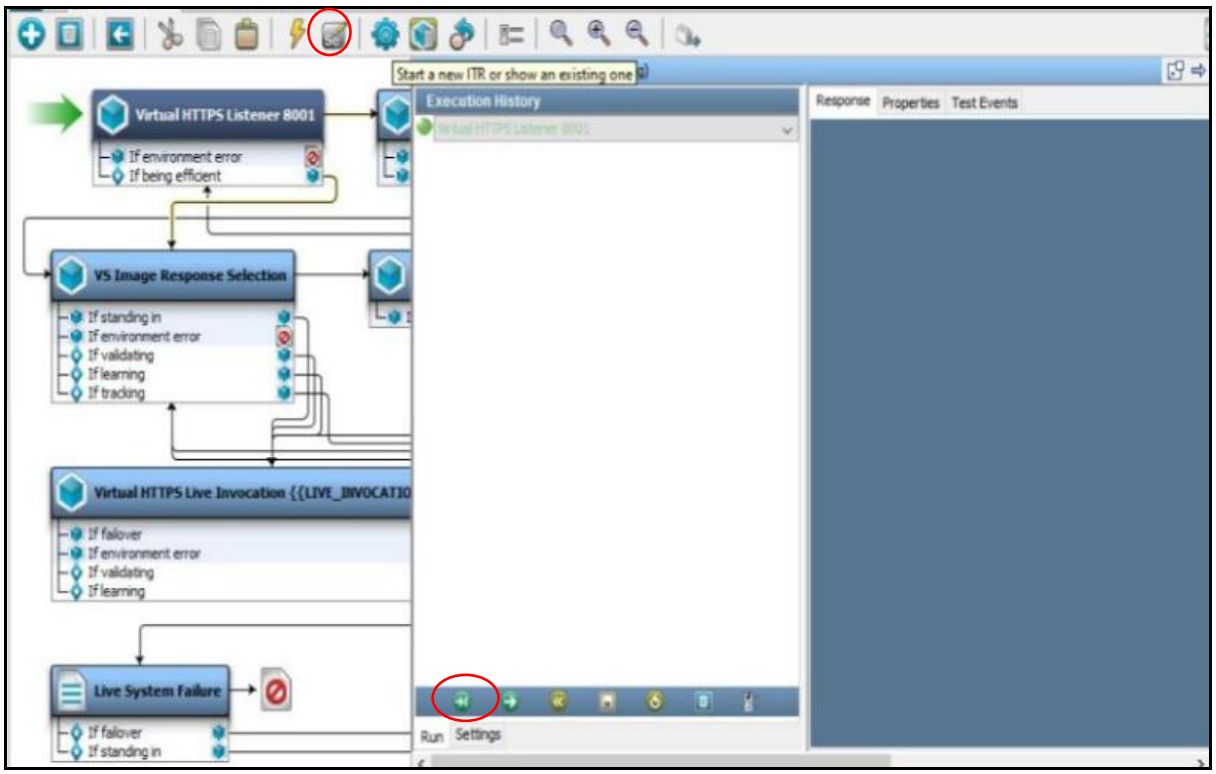
13. Enter the copied base path from VSM and click on ok



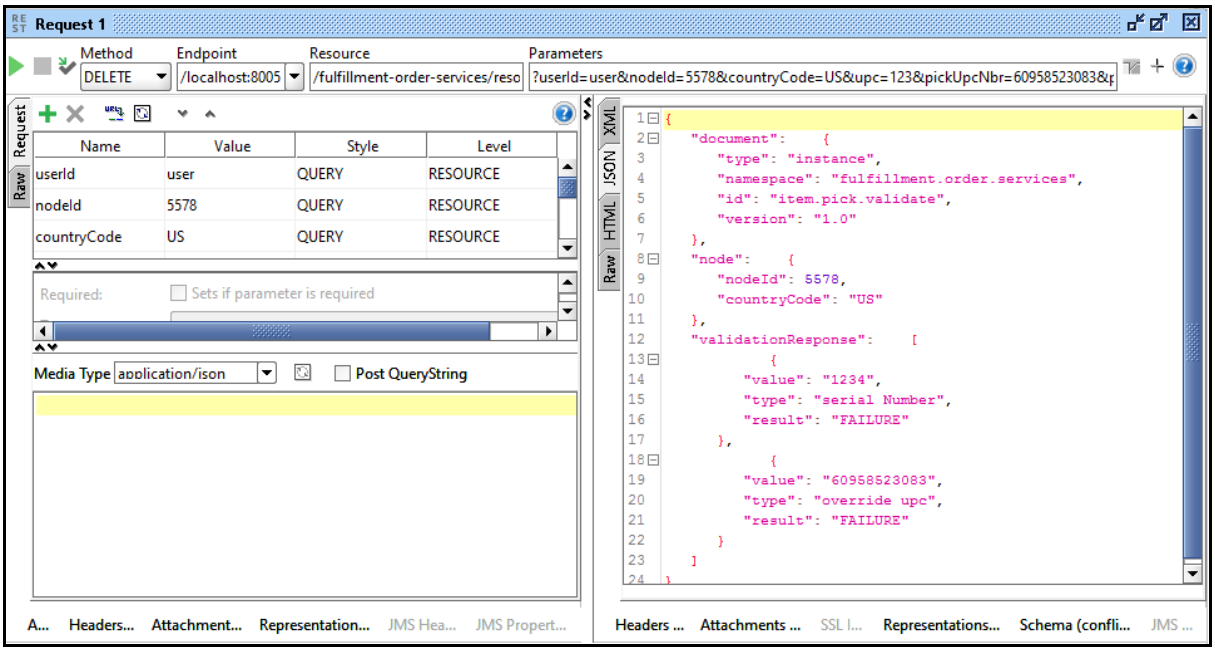
14. Enter the necessary endpoint, method, resource and parameters details
    - Endpoint: Transport protocol://target host: listener port.
    - Method: DELETE
    - Resource: Base Path
    - Parameters and their value



15. Go to VSM and start ITR and click on automatically execute the test.

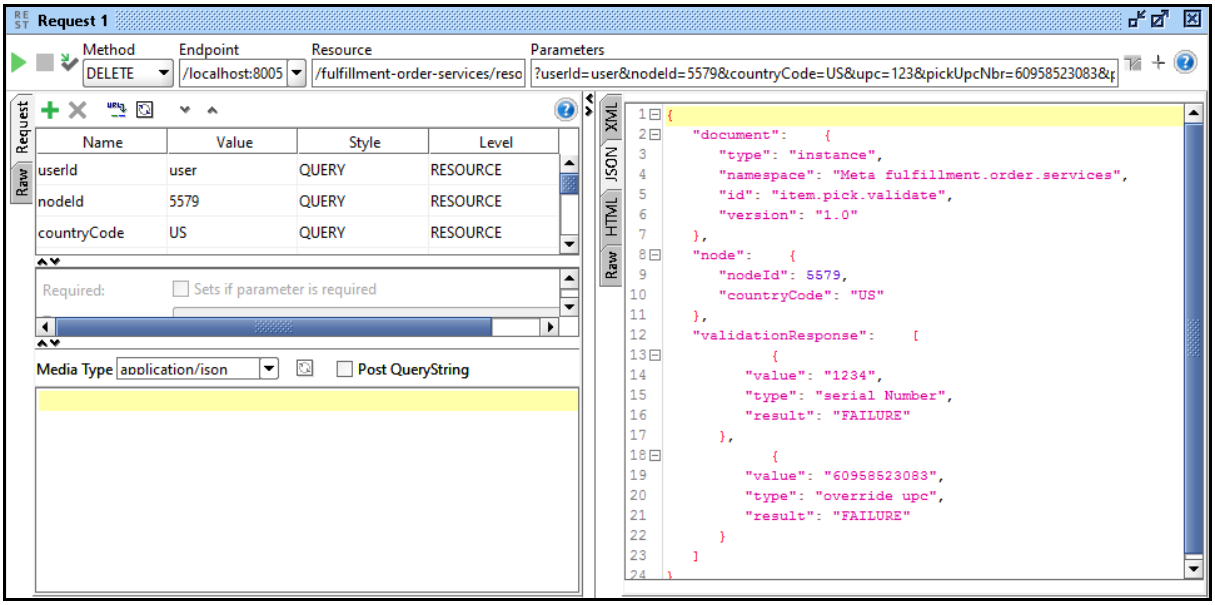


16. Click on run button in SOAP UI. It will hit the same response as in Virtual Service



17. SOAP UI will hit the META Response if current request doesn't match the actual request.

E.g.: (Actual Request nodeid-5578, Current Request node id – 5579)



Sample: Request and Response Pair files

15. Properties & Configurations

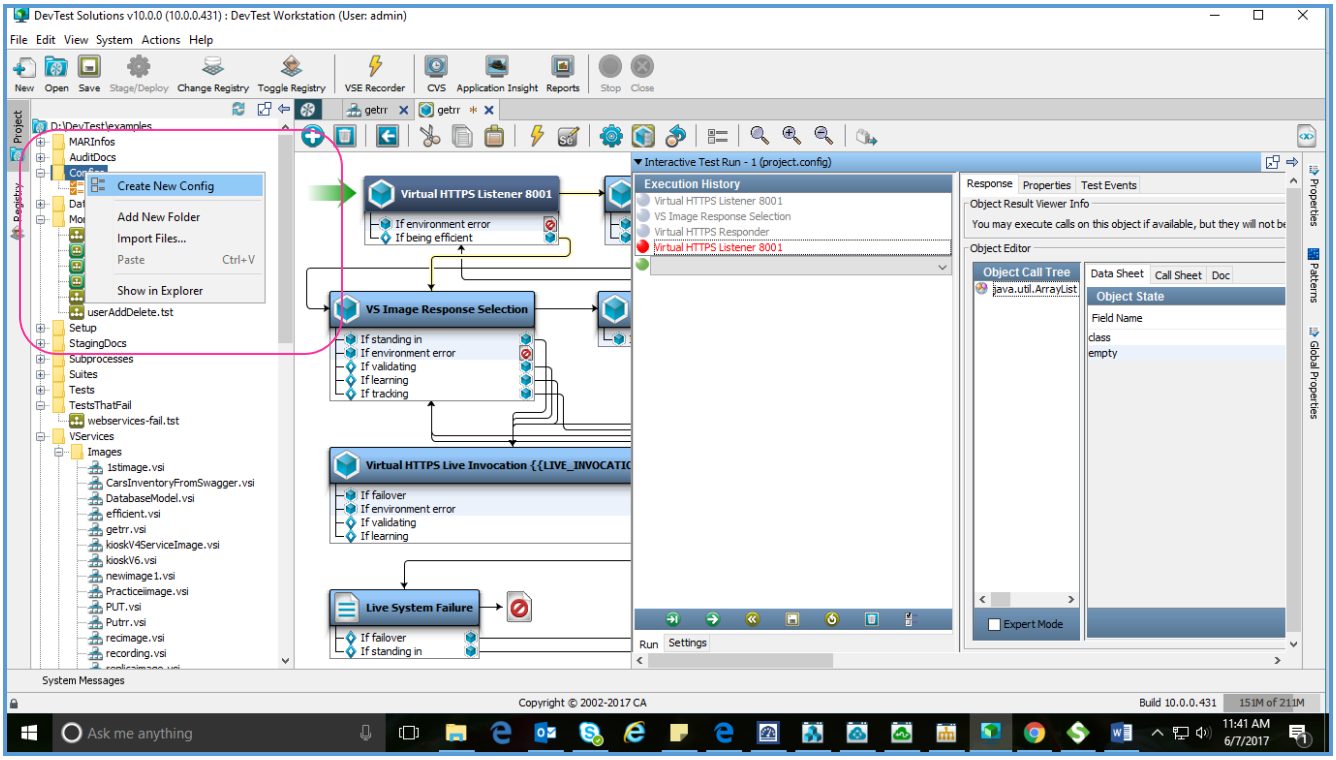
The default configuration has the name **project.config**, and is created automatically for a new project. The **project.config** file is located in the Configs folder in the Project panel. The file extension is not shown. You can add the properties to the project.config file and, if necessary can also create a configuration file.

15.1 Creation of the Configuration File

Follow below steps:

- ✓ Right-click the **Configs folder** in the **Project panel** and select **Create New Config**.
- ✓ Enter the Name of the new configuration
- ✓ Click OK button

The properties editor opens the created config file:



15.2 Adding Property to the Configuration File

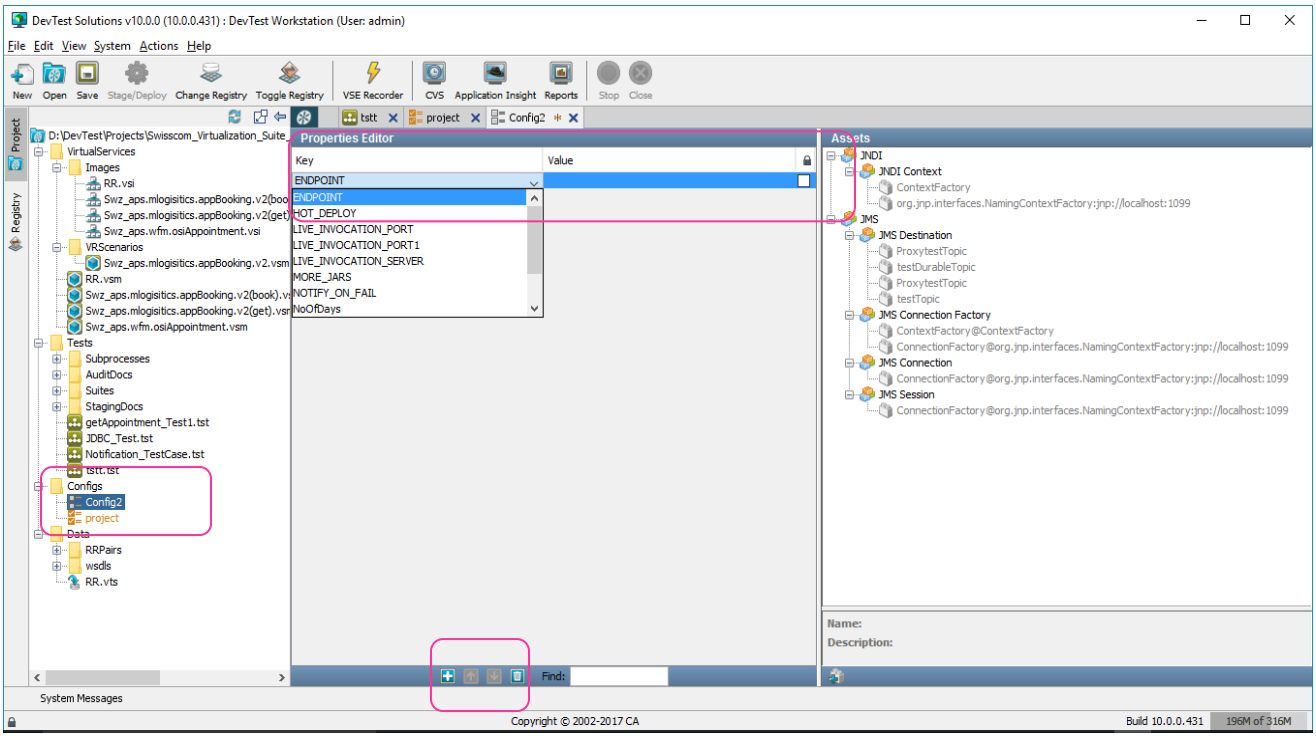
1. To add a row, click Add Icon at the bottom of the properties editor
2. Select ENDPOINT1 from the Key field
3. Enter the following value in the Value field:

   http://localhost:8001/itkoExamples/EJB3UserControlBean?wsdl

Note: The port from the test case (8080) is changed to the listen port for the VSE Recorder (8001). The recorder can intercept the request and response on the way to and from the live web service application on port 8080.

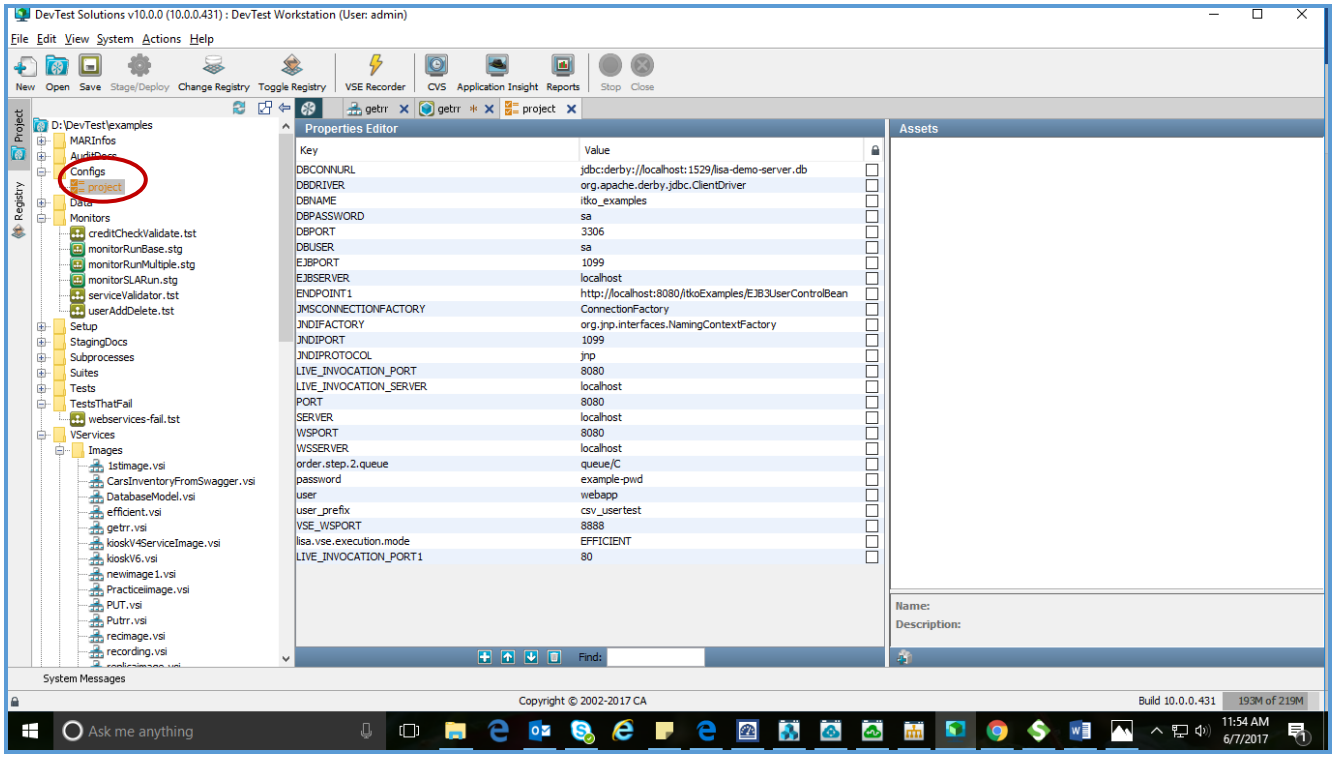
4. Click Save on the main toolbar
5. To close the config file, click X on the editor tab
6. Continue with "Step 6 - Activate Config File"

https://docops.ca.com/devtest-solutions/8-0-1/en/getting-started/ca-service-virtualization-tutorial/step-5-create-a-config-file

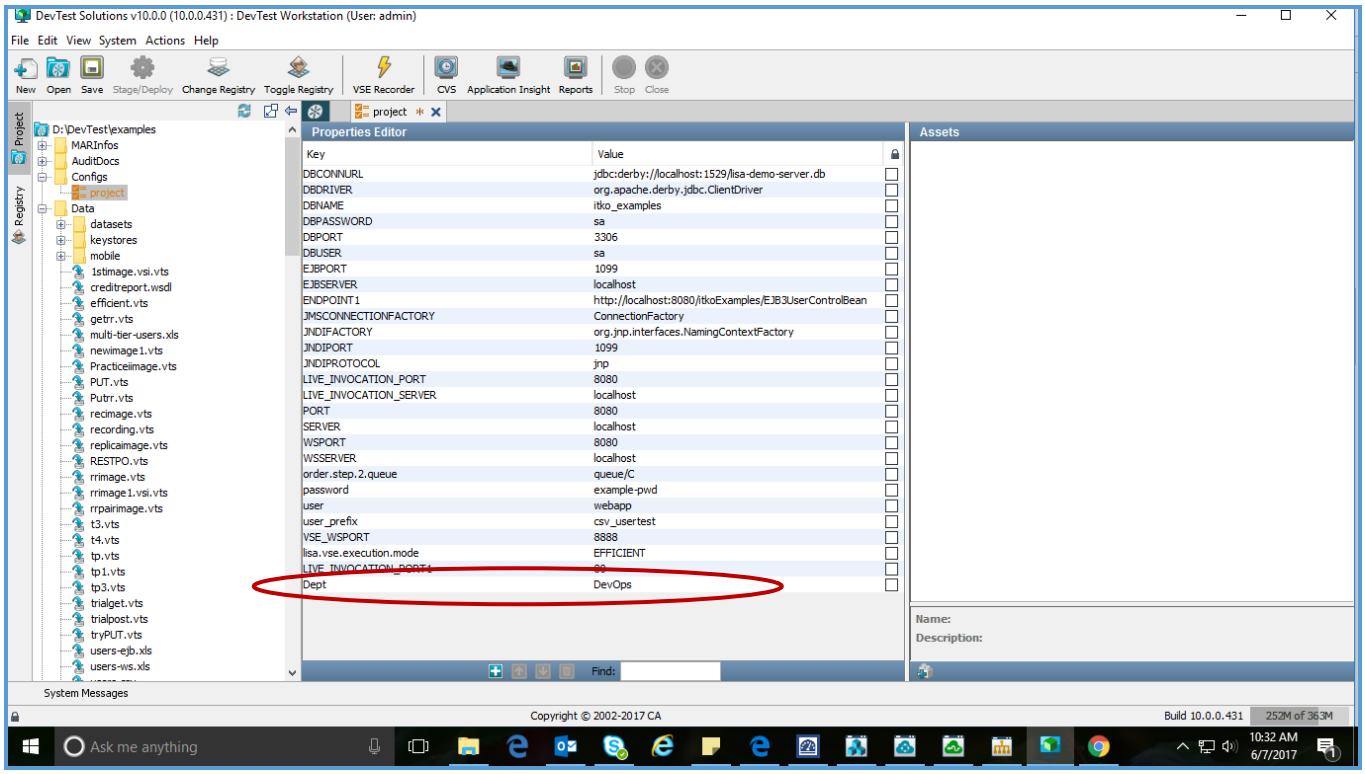


15.3 Adding Properties to Existing Configuration File

1. Double click **Config file** from project panel and property window appears.



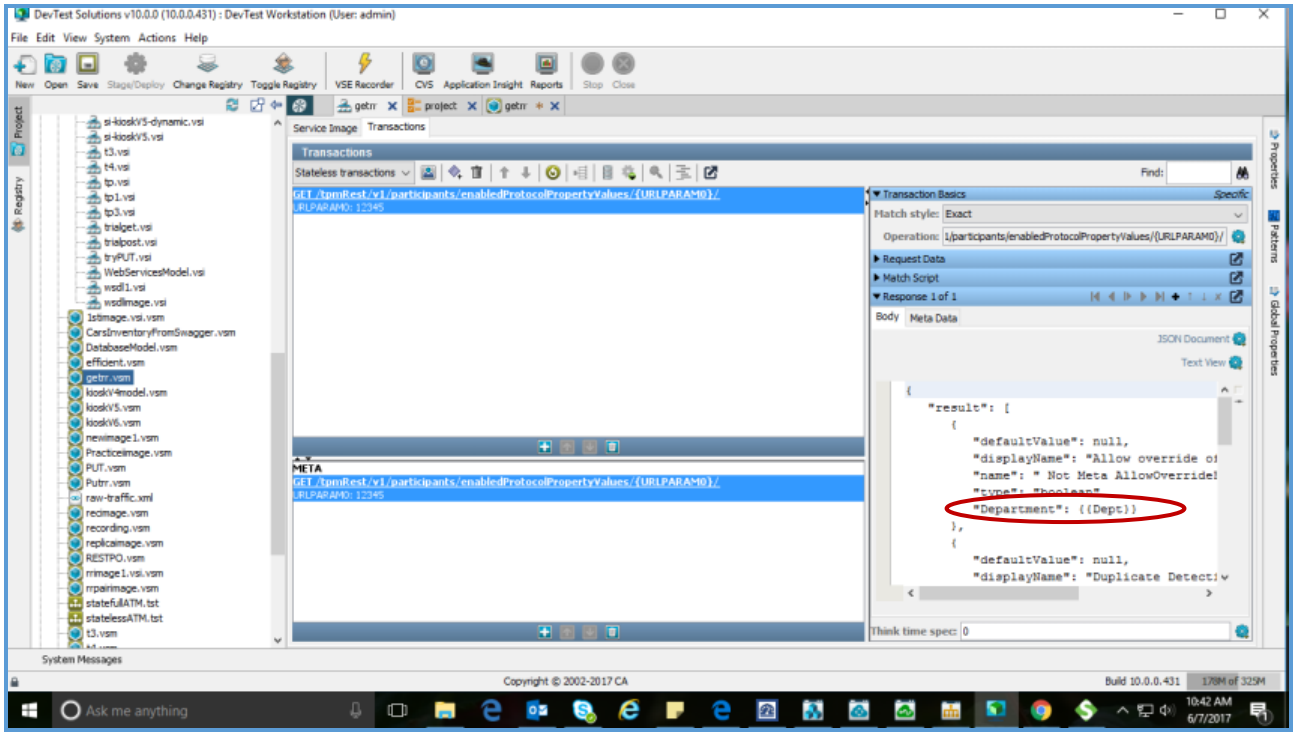
2. To add a property, click Add Icon at the bottom of the properties editor
3. Enter the property name and value. (ex: **Dept & DevOps**)



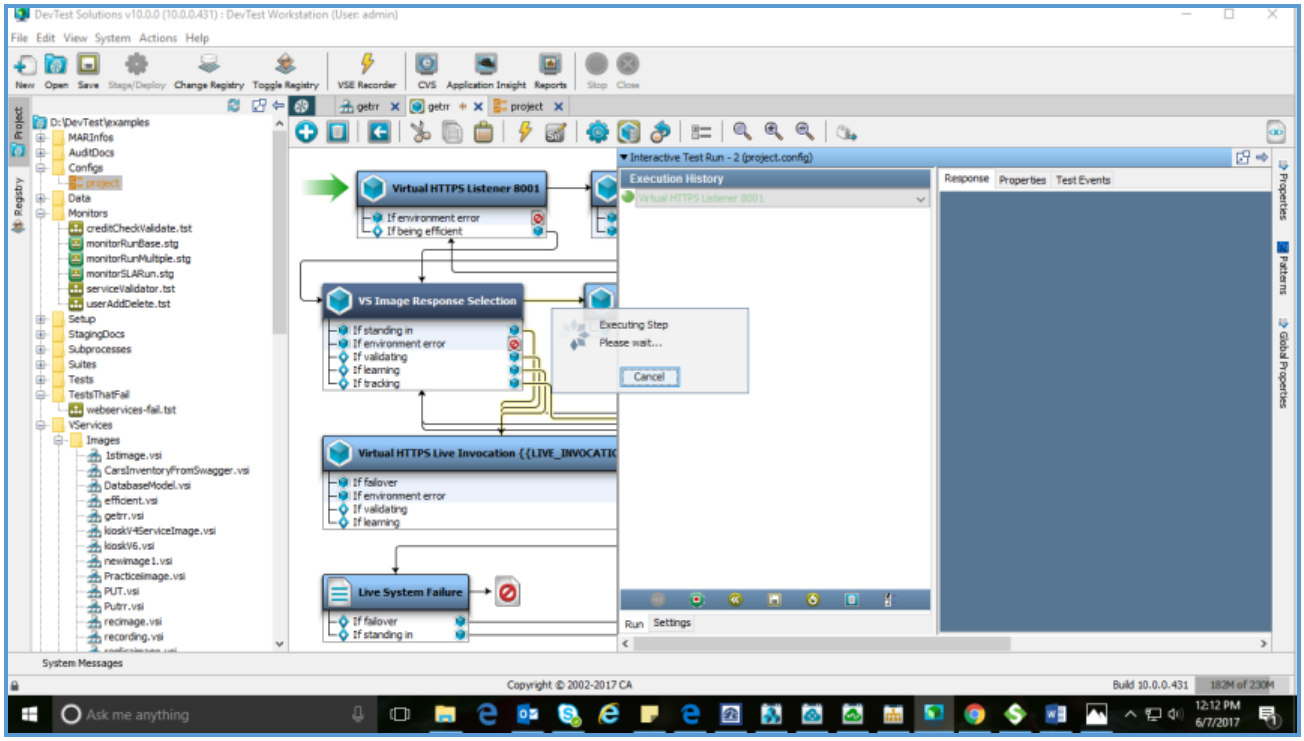
Note: Click on check box only to encrypt passwords

15.4 Implementation of the Property

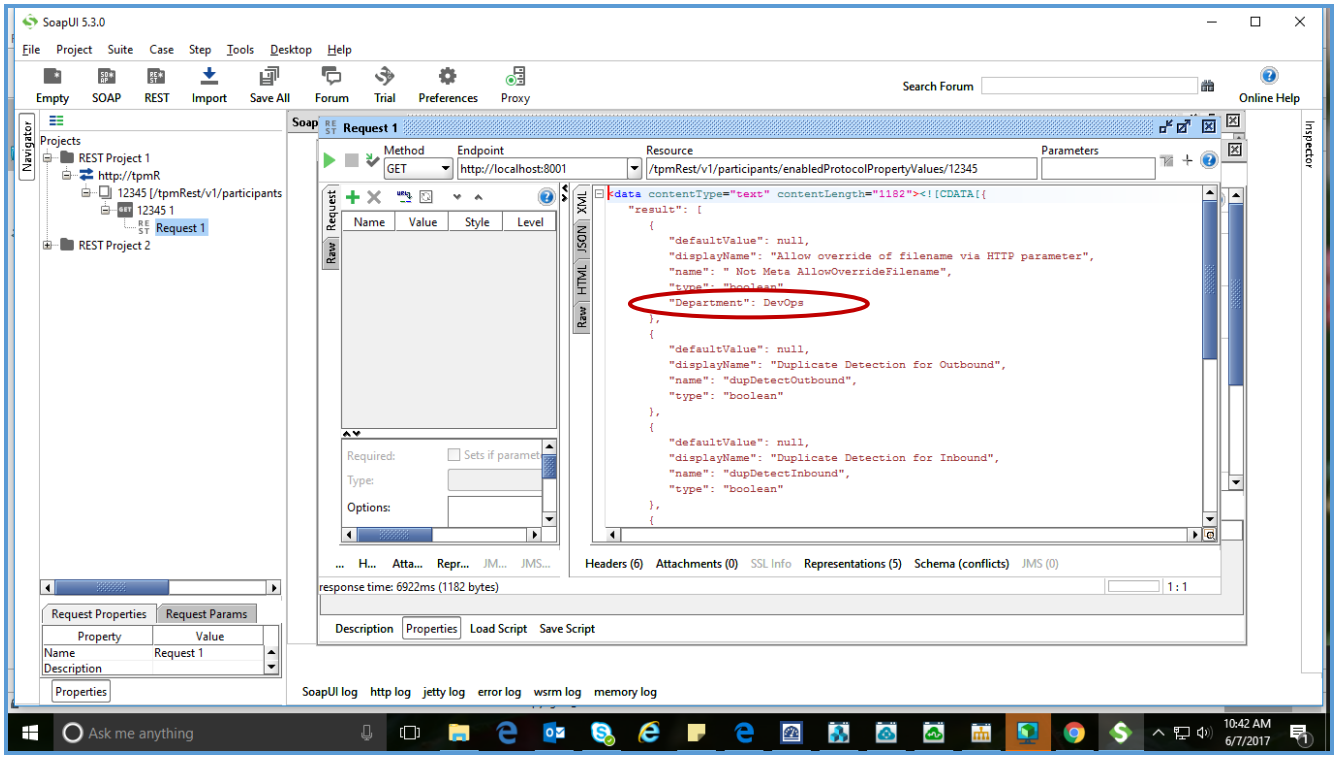
1. Open VSI and edit in the response body to add the property that is defined in config file and save the VSM



2. Run it in ITR mode

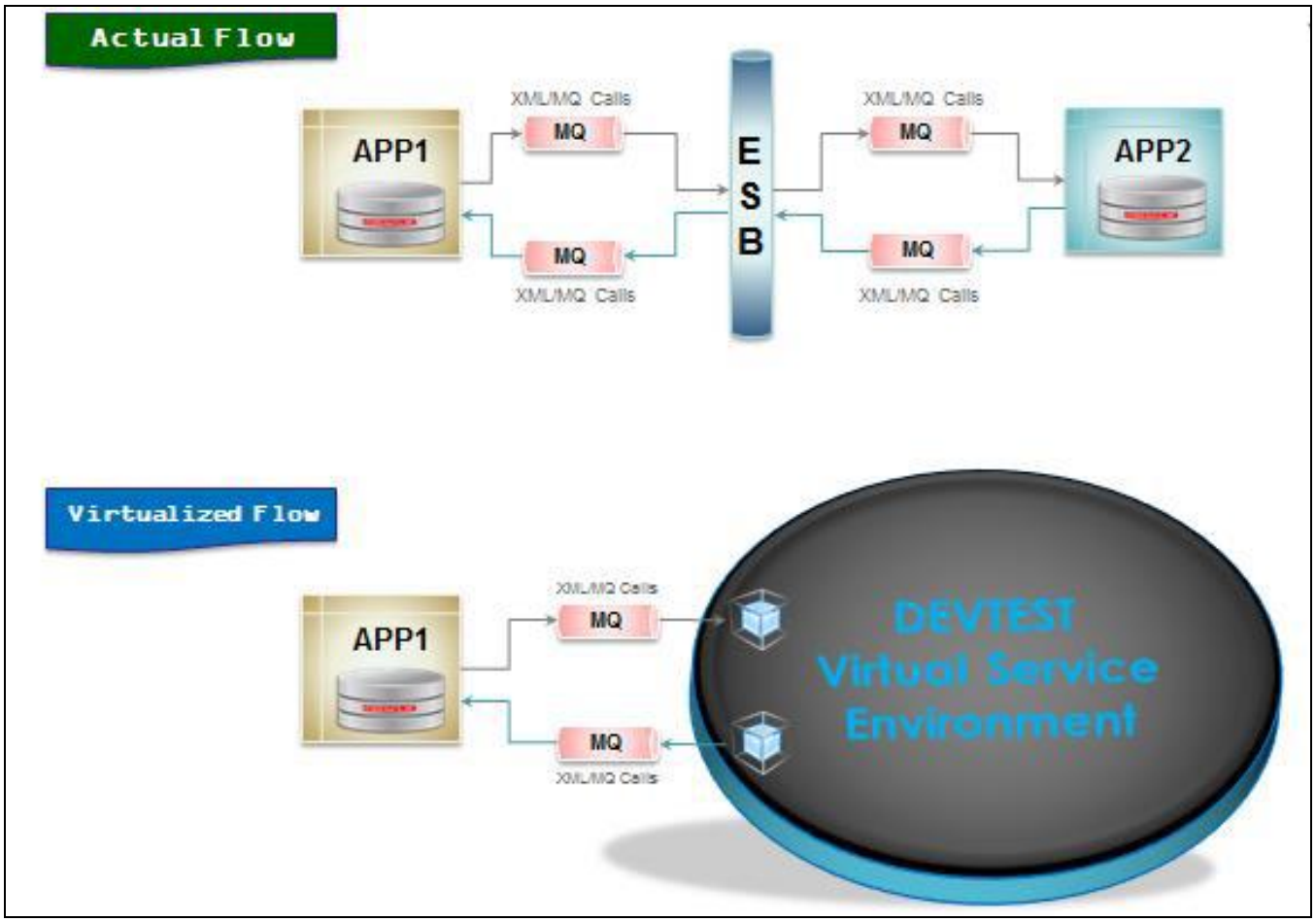


3. To check this property is implemented successfully or not run this in SOAP UI. The property is added successfully.



16. MQ topic virtualization

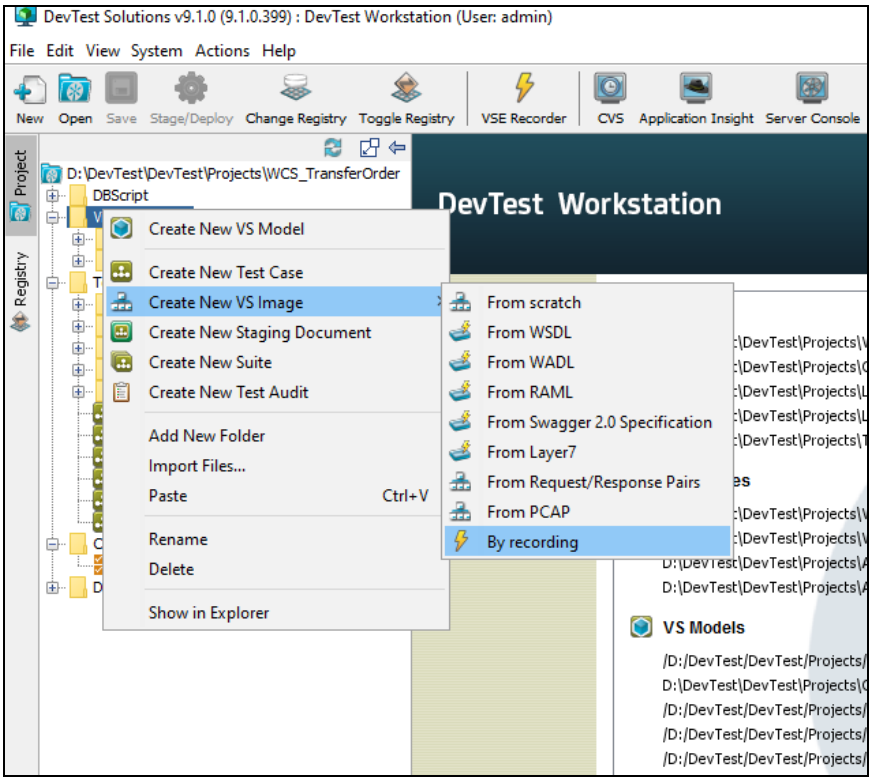
For MQ virtualization in CA Dev Test (LISA) it requires us to have dedicated proxy queues on queue manager for us to virtualize and playback the service.



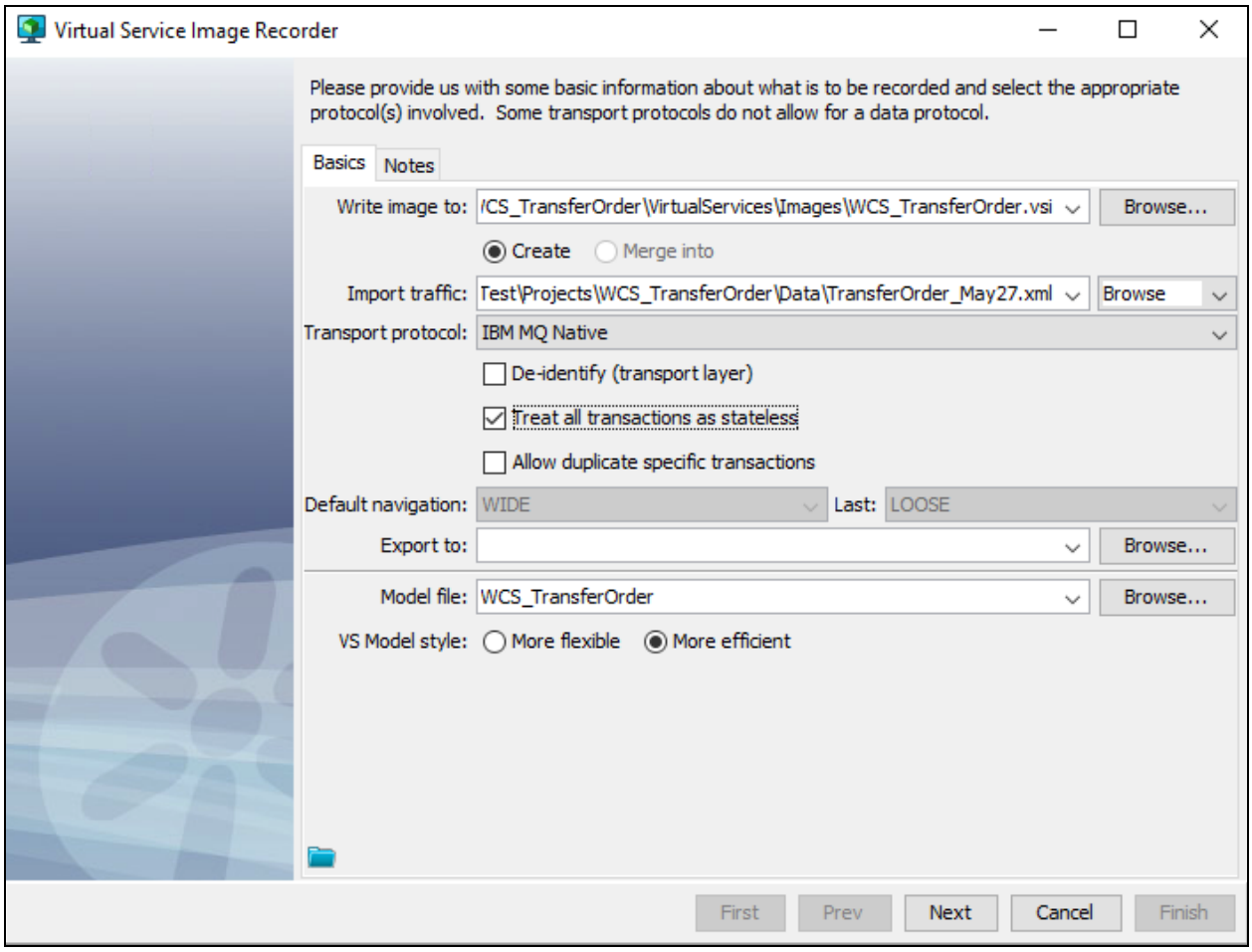
16.1 Creation of MQ Services

Follow below steps to create:

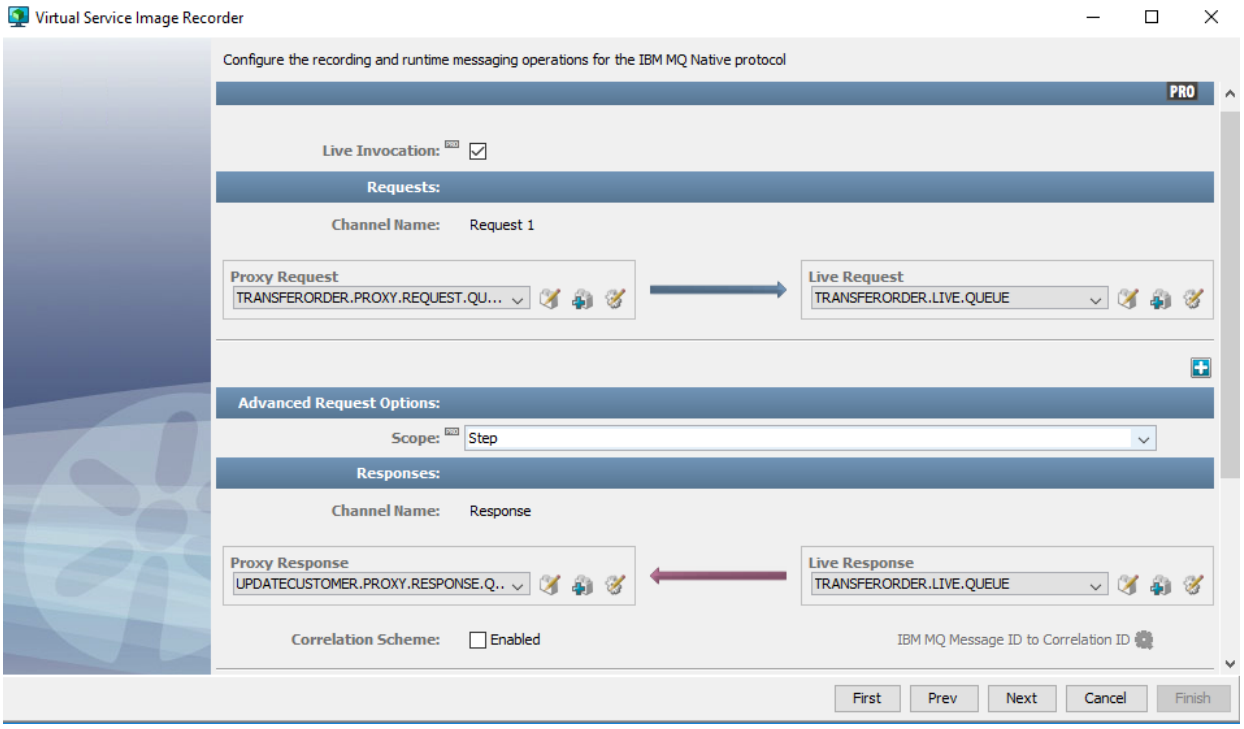
1. Right-click the VirtualServices node on the Project panel and select Create a VS Image, by Recording



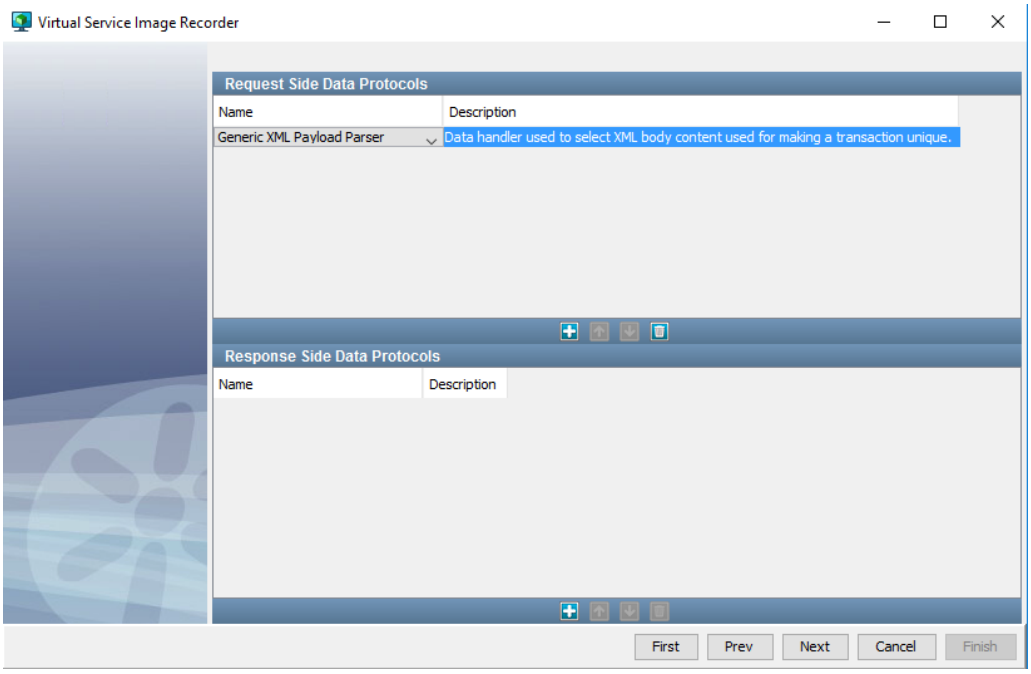
2. Provide Required details in VSI Recorder screen and click Next button

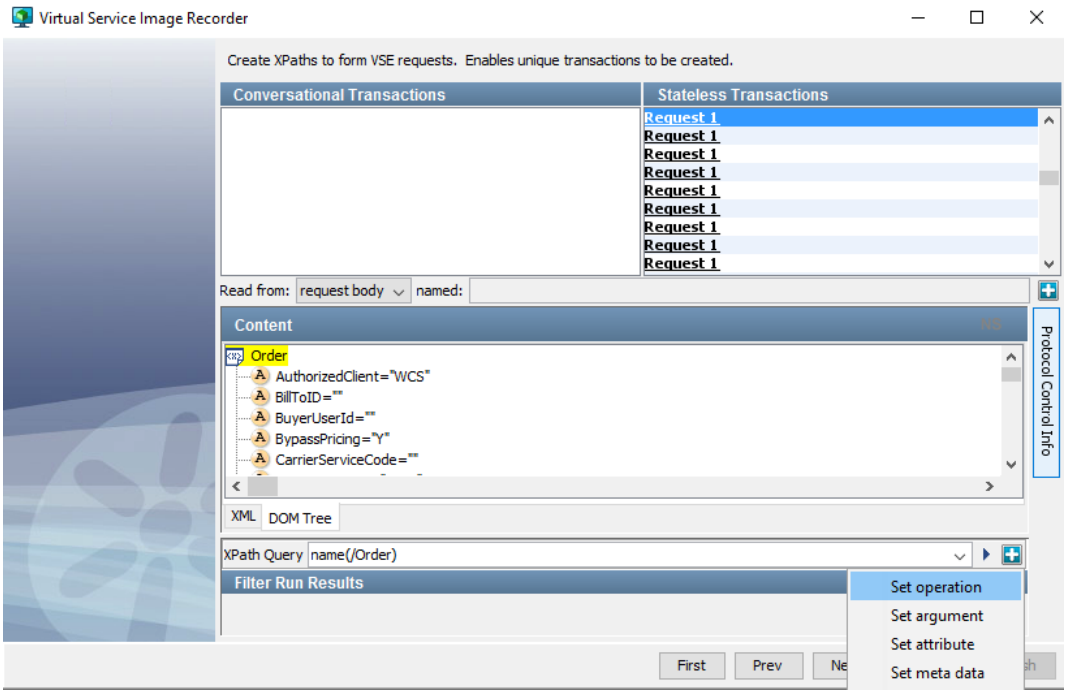


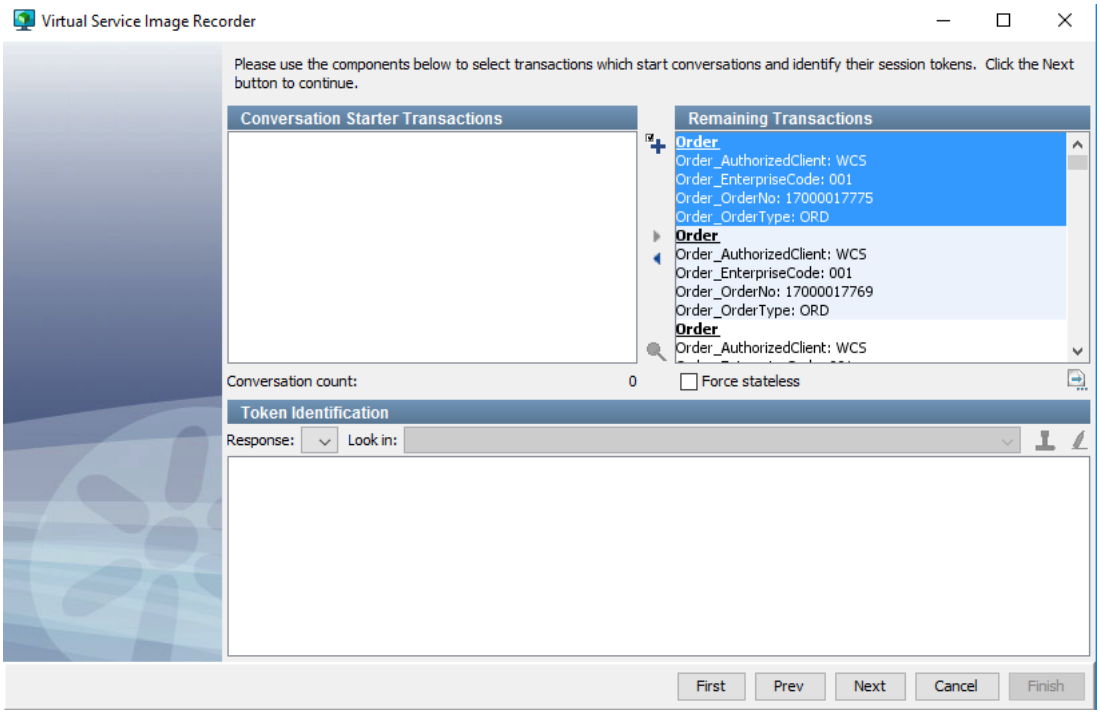
3. Provide details of all 4 Queues

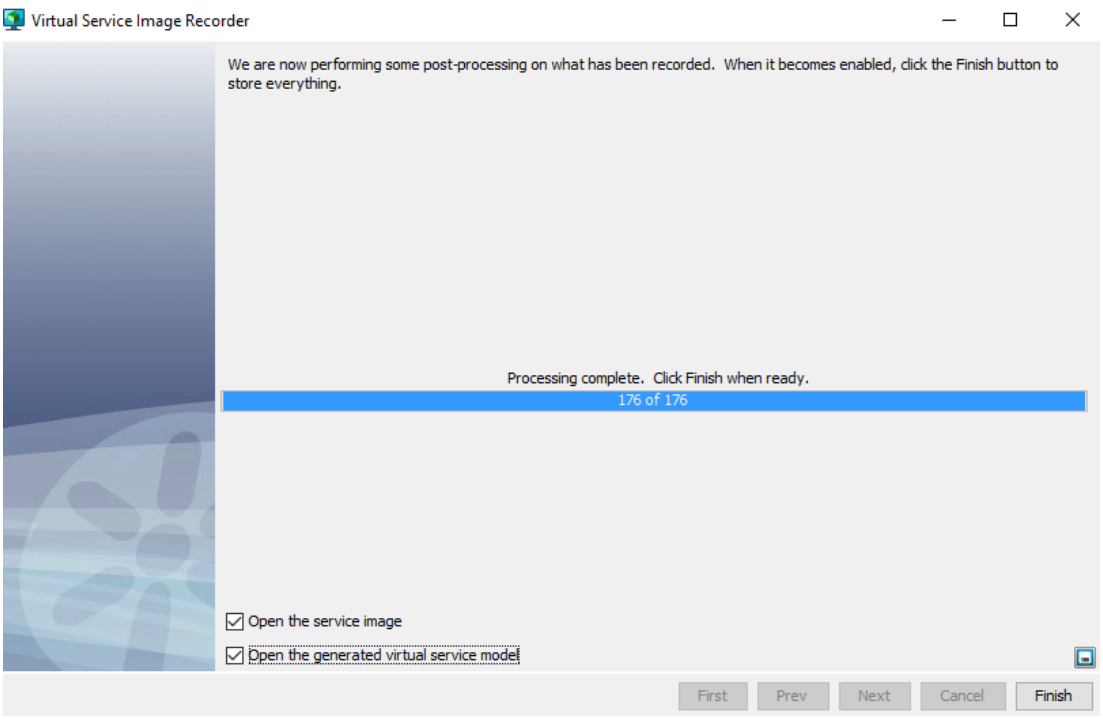


4. Select appropriate data protocol and complete VSI, VSAM creation process




Virtual Service Image Recorder
Create XPaths to form VSE requests. Enables unique transactions to be created.
Conversational Transactions
Stateless Transactions
Request 1
Request 1
Request 1
Request 1
Request 1
Request 1
Request 1
Request 1
Request 1
Read from: request body named:
Content
Order
AuthorizedClient="WCS"
BillToID=""
BuyerUserId=""
BypassPricing="Y"
CarrierServiceCode=""
Protocol Control Info
XML DOM Tree
XPath Query name(/Order)
Filter Run Results
Set operation
Set argument
Set attribute
Set meta data
First
Prev


Virtual Service Image Recorder
Please use the components below to select transactions which start conversations and identify their session tokens. Click the Next button to continue.
Conversation Starter Transactions
Remaining Transactions
Order
Order_AuthorizedClient: WCS
Order_EnterpriseCode: 001
Order_OrderNo: 17000017775
Order_OrderType: ORD
Order
Order_AuthorizedClient: WCS
Order_EnterpriseCode: 001
Order_OrderNo: 17000017769
Order_OrderType: ORD
Order
Order_AuthorizedClient: WCS
Conversation count: 0
Force stateless
Token Identification
Response: Look in:
First
Prev
Next
Cancel
Finish


Virtual Service Image Recorder
We are now performing some post-processing on what has been recorded. When it becomes enabled, click the Finish button to store everything.
Processing complete. Click Finish when ready.
176 of 176
Open the service image
Open the generated virtual service model
First
Prev
Next
Cancel
Finish

17. JMS

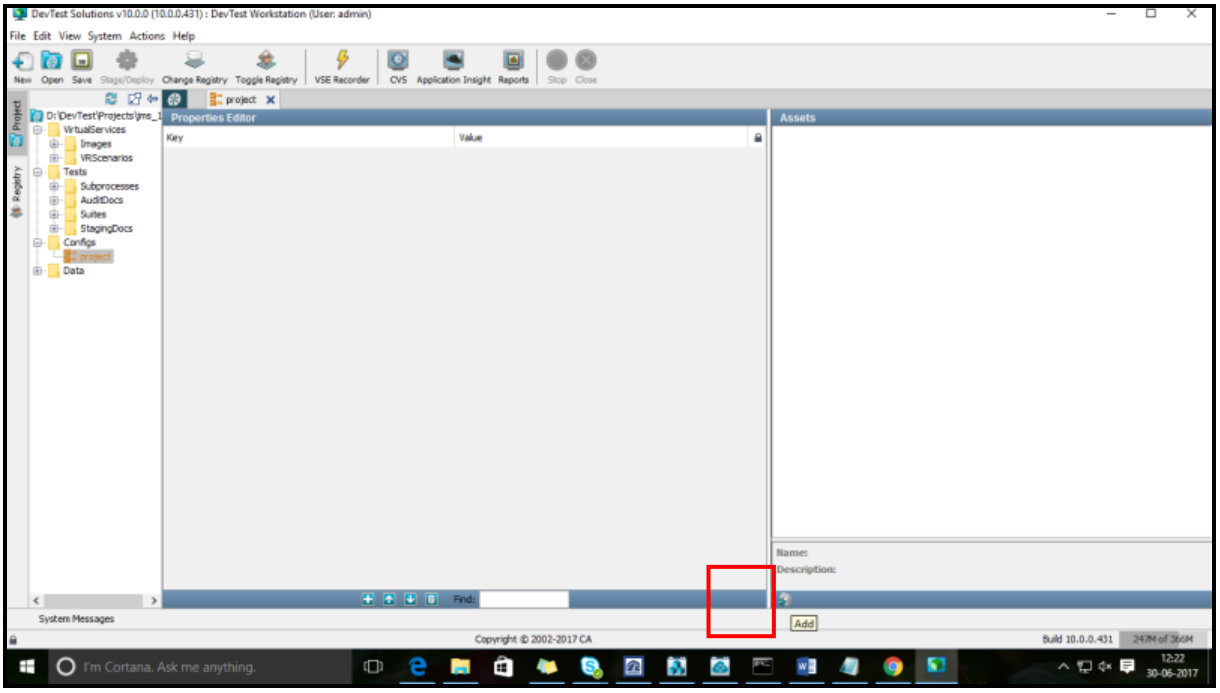
Please follow the below mentioned pre requisites for working with JMS Topic and Queues in CA DevTest:

- Make sure you have demo server installed (If not then you need to install it again if you want to use the available Topic and Queues of the Demo Server)
- Run the demo server
- Then open the link http://<hostname>:8080 in any of the browser. You will be able to see the JBoss Application page
- In the left side of the page we have a link named – JMX console
- Click on it and you will be directed to next page where you can see the available Topic's and Queues of the demo server
- You can use these Topic and Queues for JMS virtualization in your CA Devtest tool
- Here we can create our own Topic and Queues by making the following change in the mentioned file - *DevTest\DemoServer\lisa-demo-server\jboss\server\default\deploy\jms**jbossmq-destinations-service***
- If you don't want to use the Demo Server Topic and Queue, then make sure that you have a JMS server where the Topic and Queues are pre created and connection details are available beforehand.
- Place this XML file under the DemoServer folder: DevTest\DemoServer\lisa-demo-server\jboss\server\default\deploy\jms.

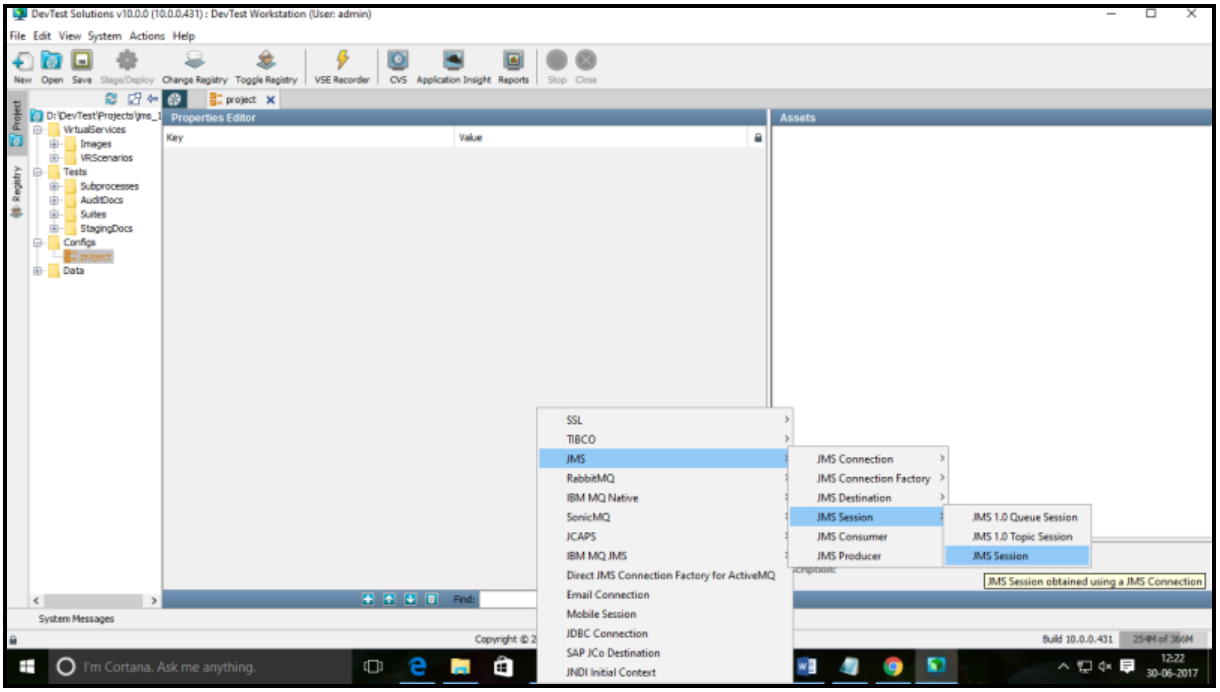
jbossmq-destinations-service.xml

Follow below steps for JMS Connection:

1. Create a New project.
2. Go to Configs > projects > click on  in Assets.

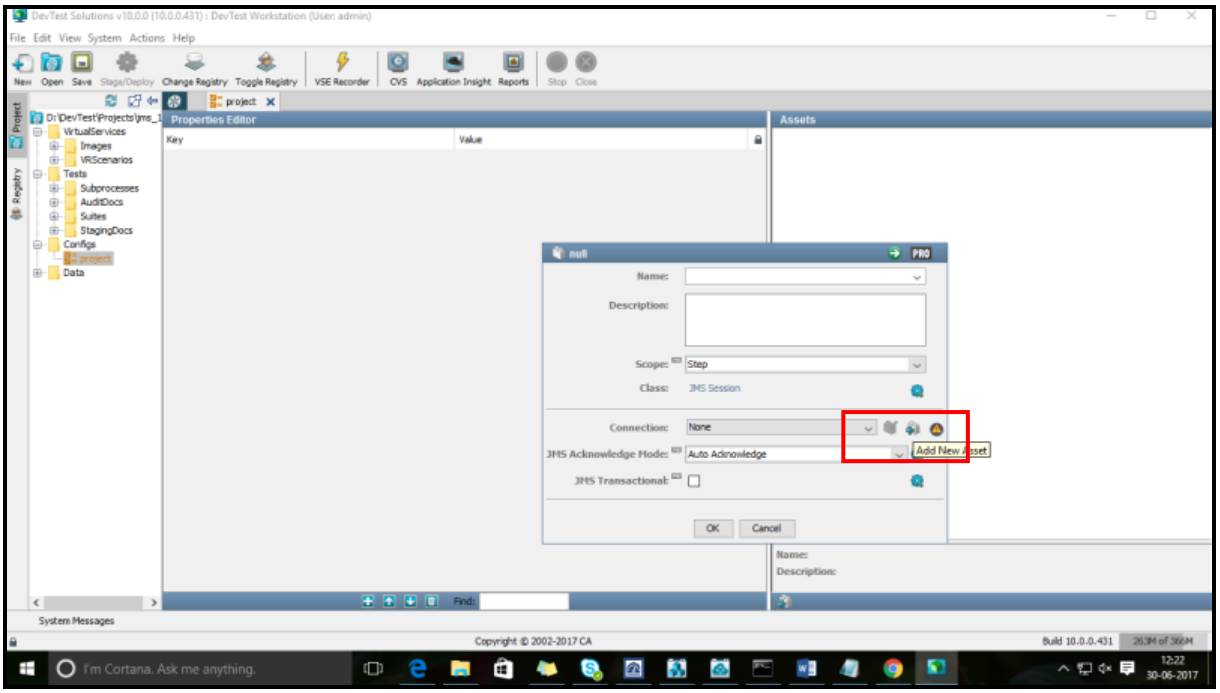


3. JMS → JMS Session → JMS Session

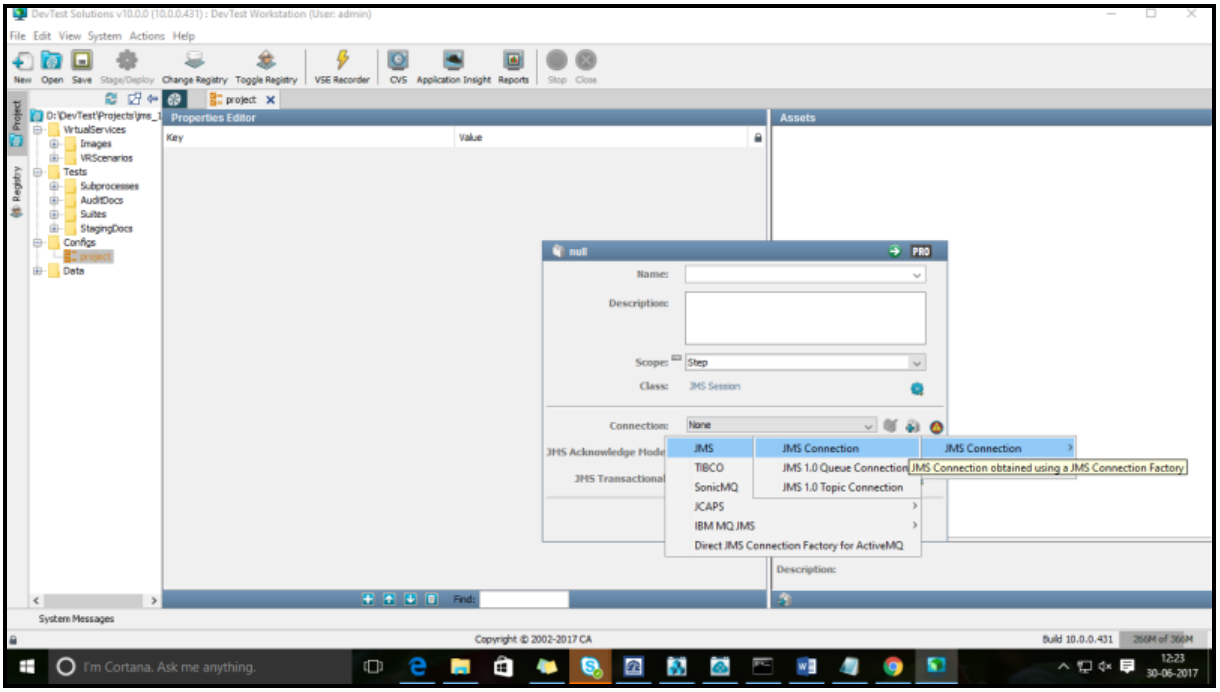


Now the key challenge is to make the connection string by connecting various links. So that the connection is successfully established.

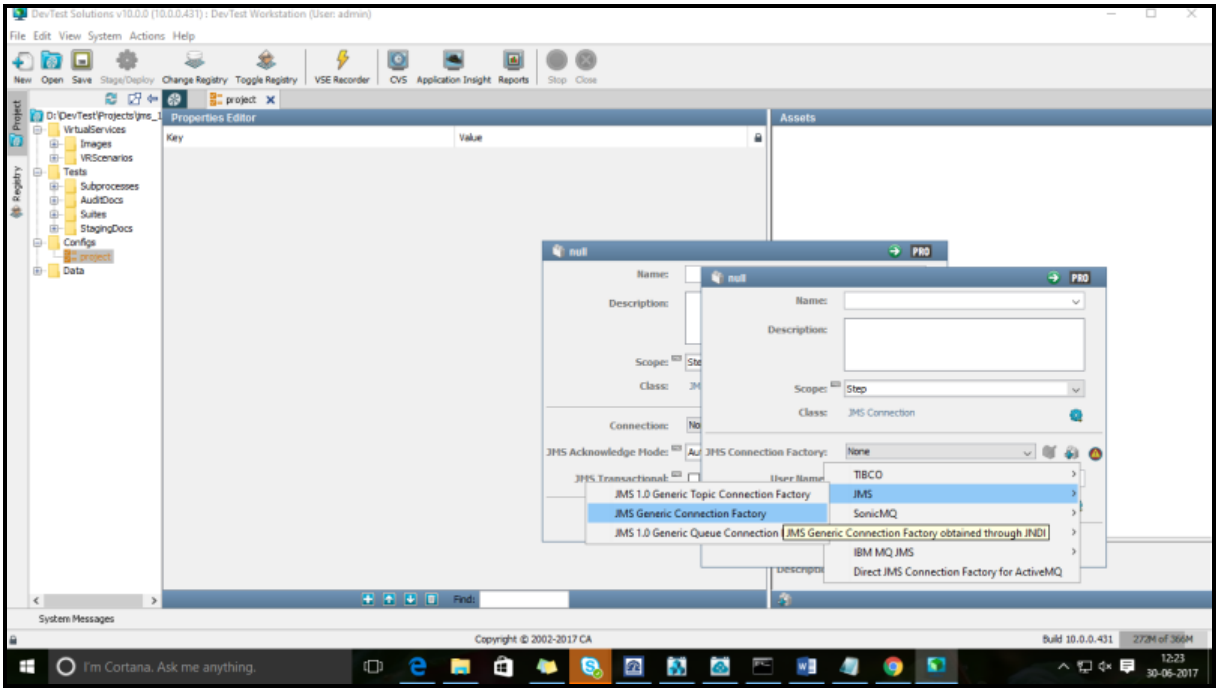
4. On clicking JMS Session, the below window appears. Click on the sign . You will be redirected to another window.



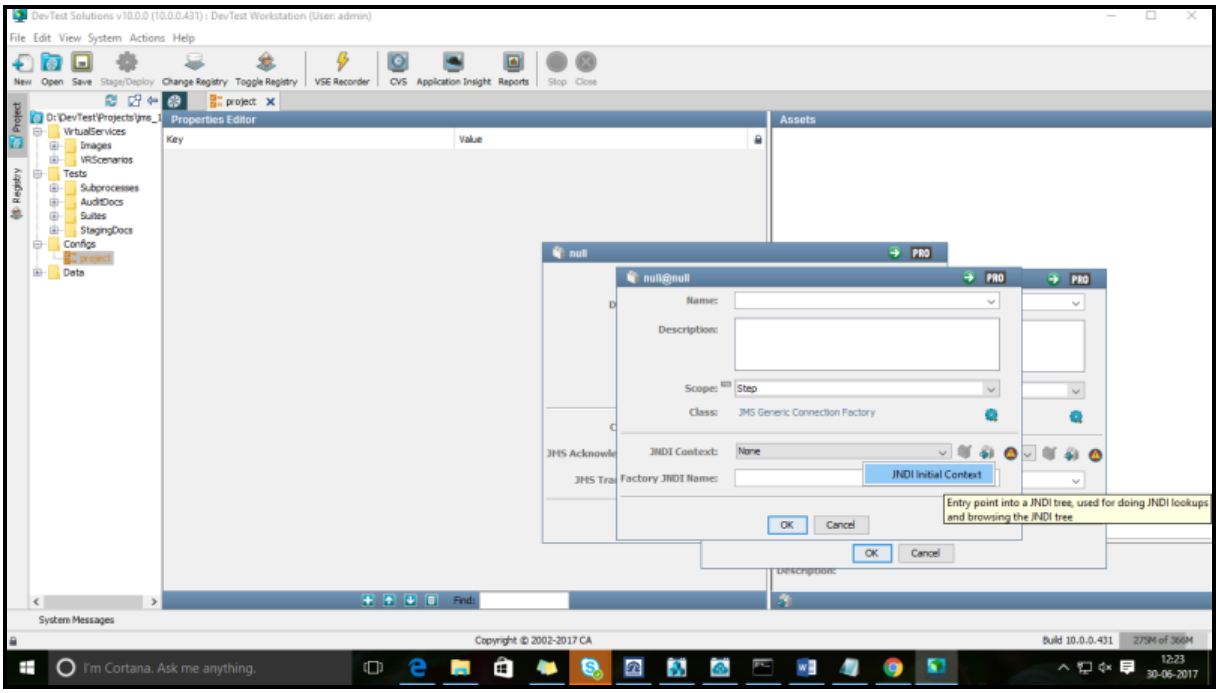
5. JMS > JMS Connection > JMS Connection.



6. Click on the  sign . You will be redirected to another window. JMS > JMS Generic Connection Factory.

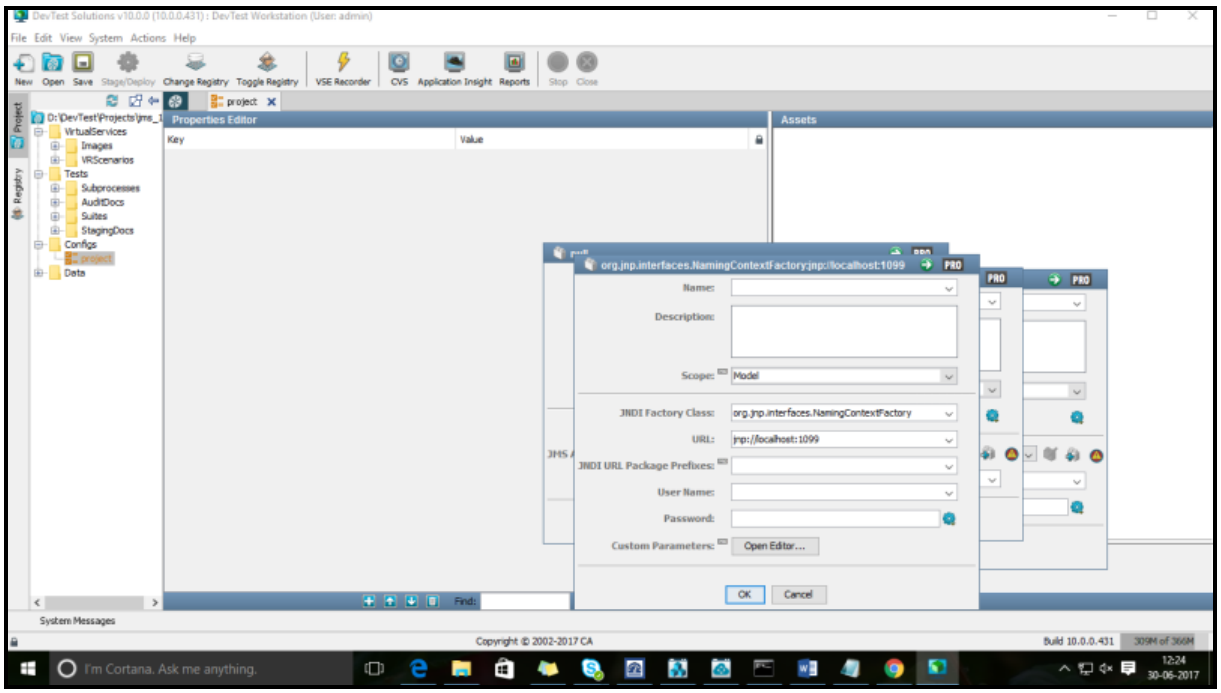


7. Click on the  sign . You will be redirected to another window. Select JNDI Initial Context and OK.

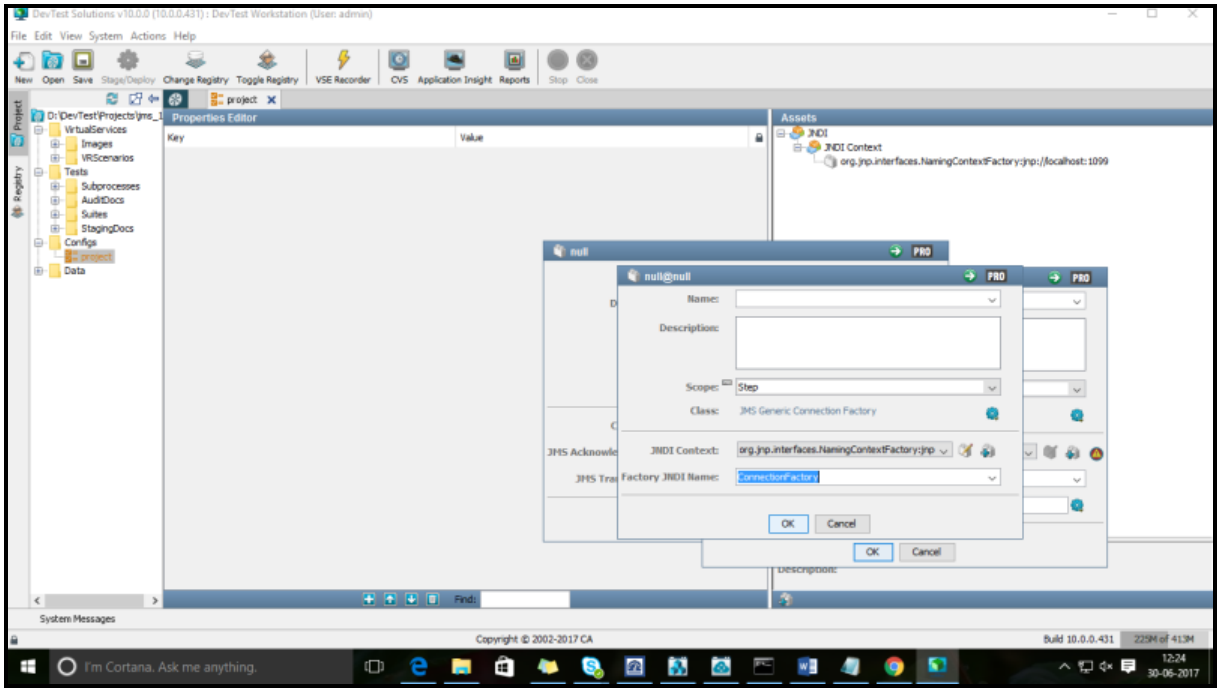


8. Enter the details and click on ok.

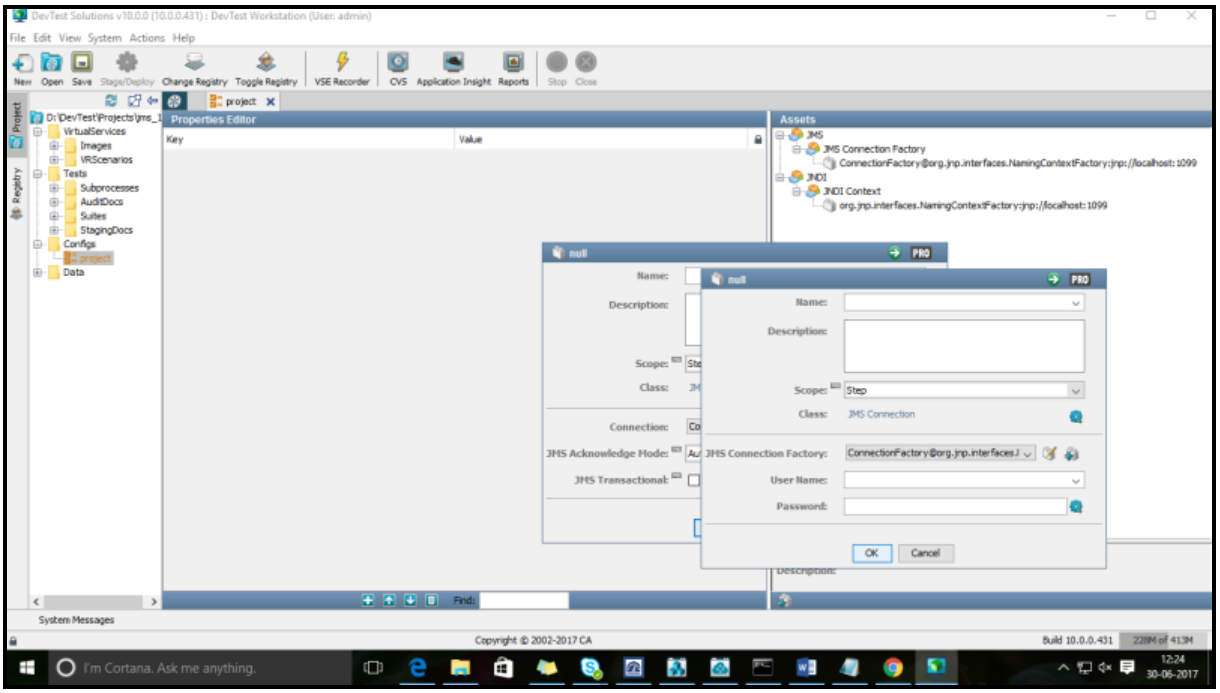
    - JNDI Factory class: org.jnp.interfaces.NamingContextFactory.
    - URL: jnp://localhost:1099



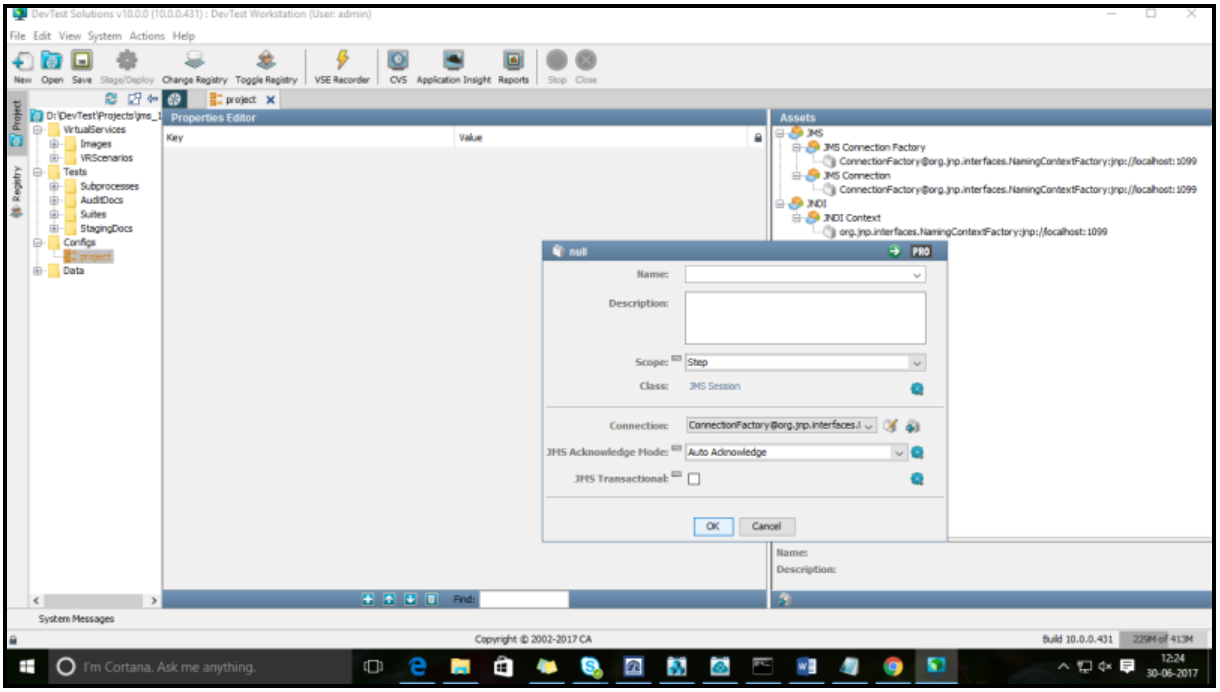
9. In Factory JNDI Name: ConnectionFactory and ok.



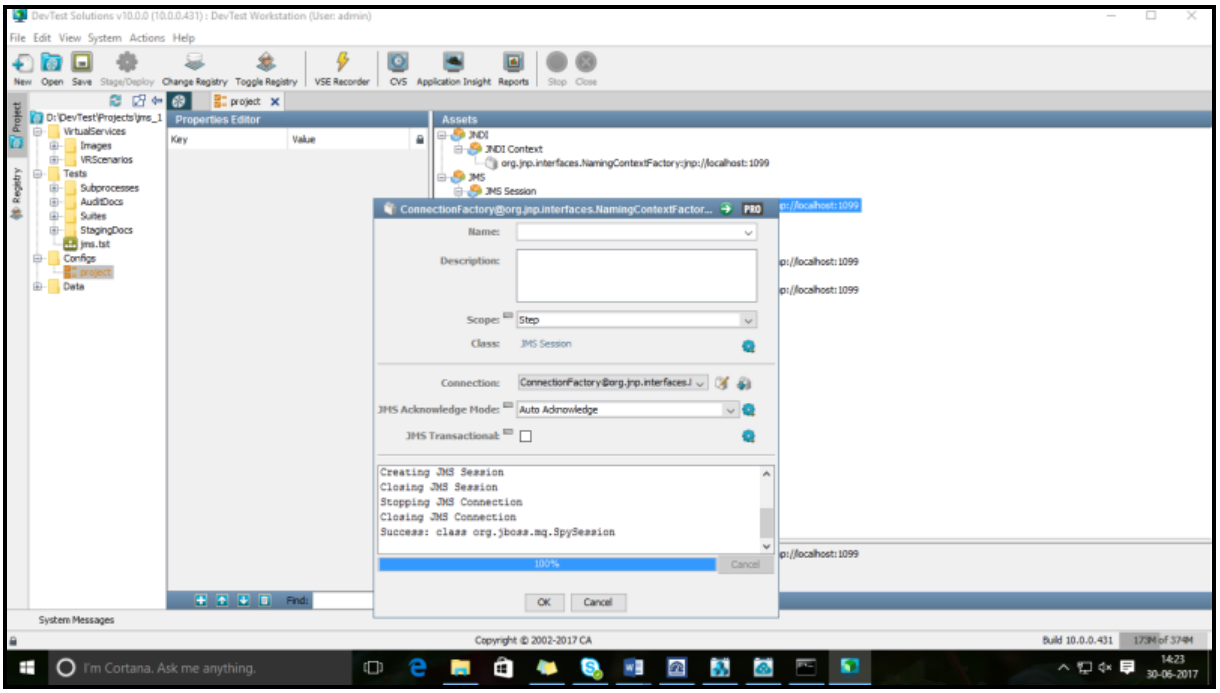
10. Click ok.



11. Then click on the  sign which appears on the below figure to test whether the connection is successfully getting established or not.



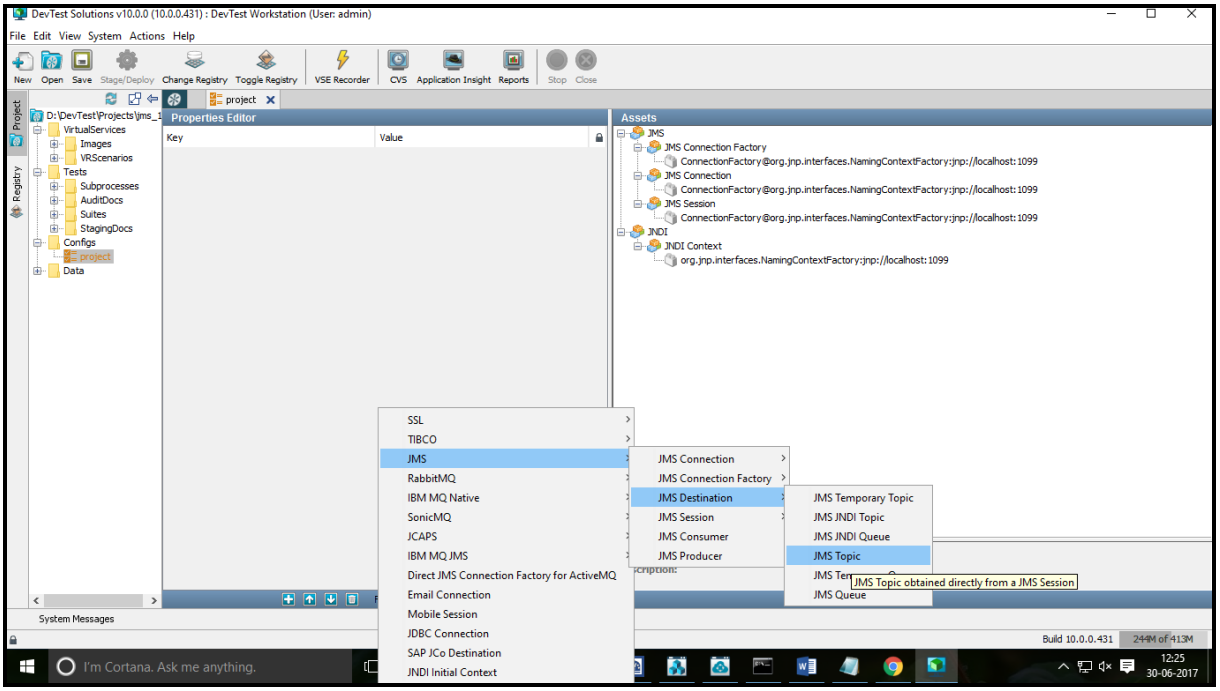
12. When connection is successfully established you will get below message.



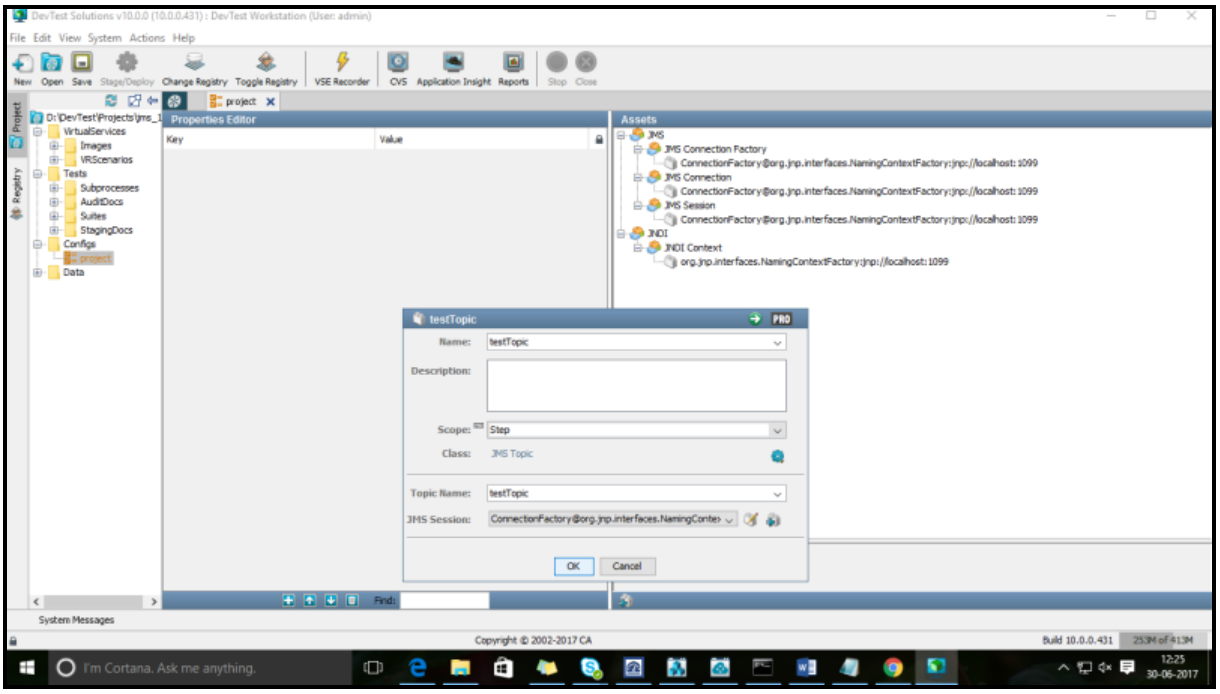
17.1 How to connect with JMS Topic and Queues

Once we have established the connection with the JMS Server. The next thing is to connect the required JMS Topic and JMS Queue. You can select any of the options which you want or required i.e. Topic or Queue.

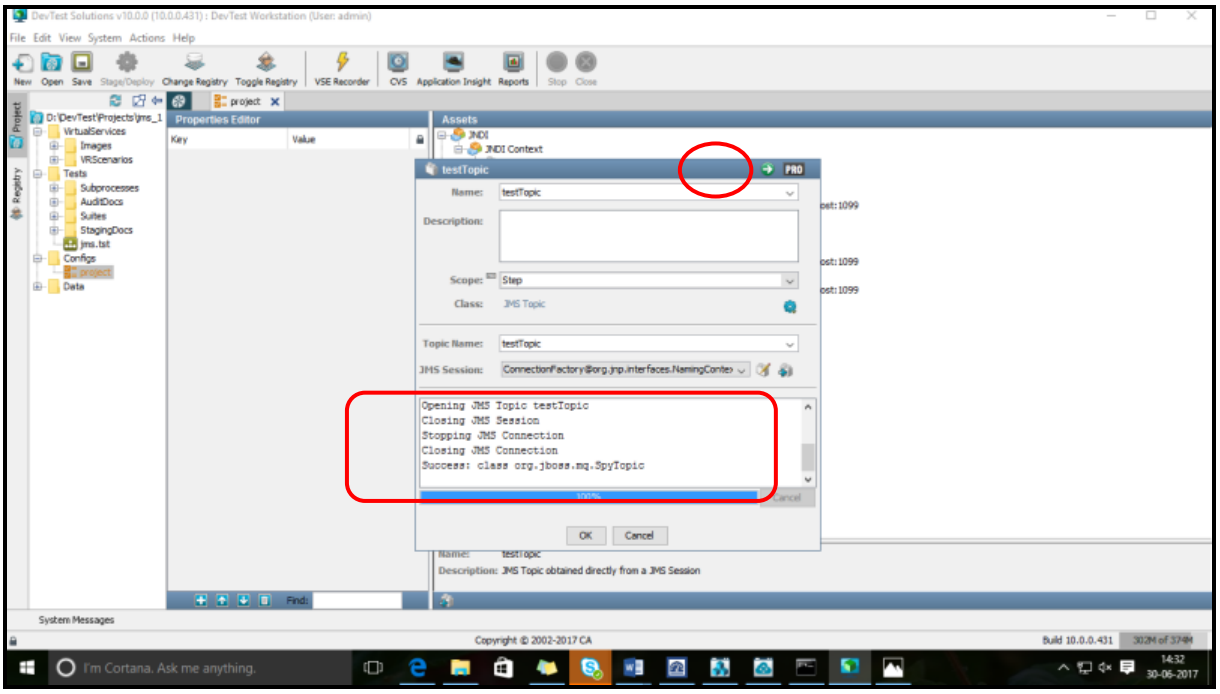
1. Go to Configs > projects > Assets > JMS > JMS Destination > JMS Topic



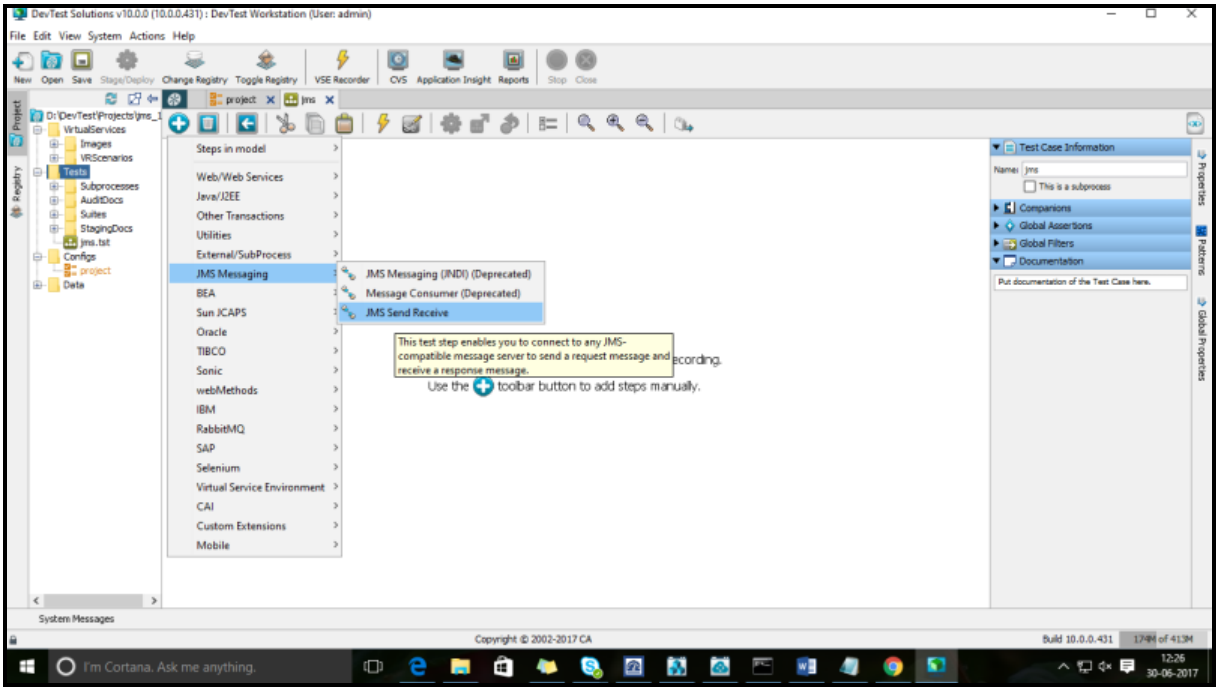
2. Provide name of the Topic which you want to make connection in the text box next to Topic Name



3. Test whether we are available to established connection to that particular Topic by clicking on  button.

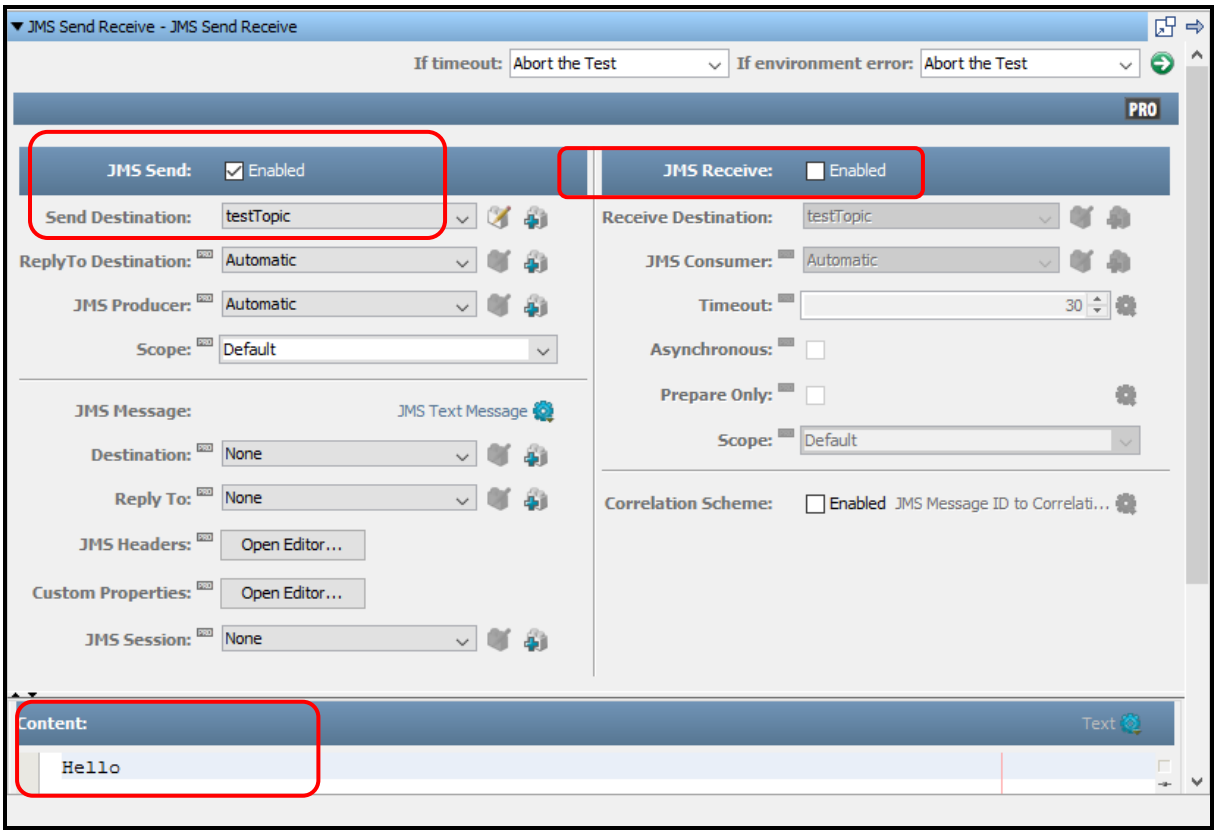


4. Create new Test Case >ADD JMS Messaging Step > JMS Send Receive

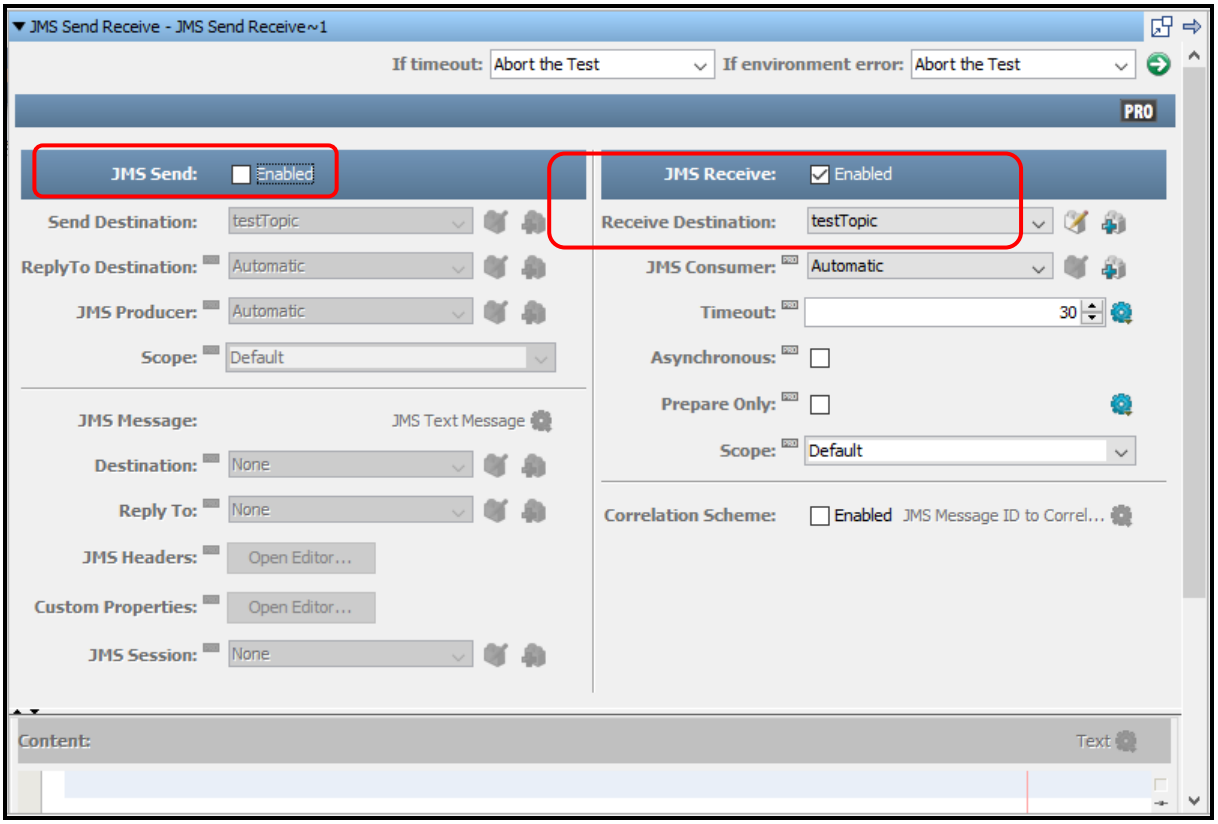


5. Select the topic created earlier in send destination. Write in Content area.

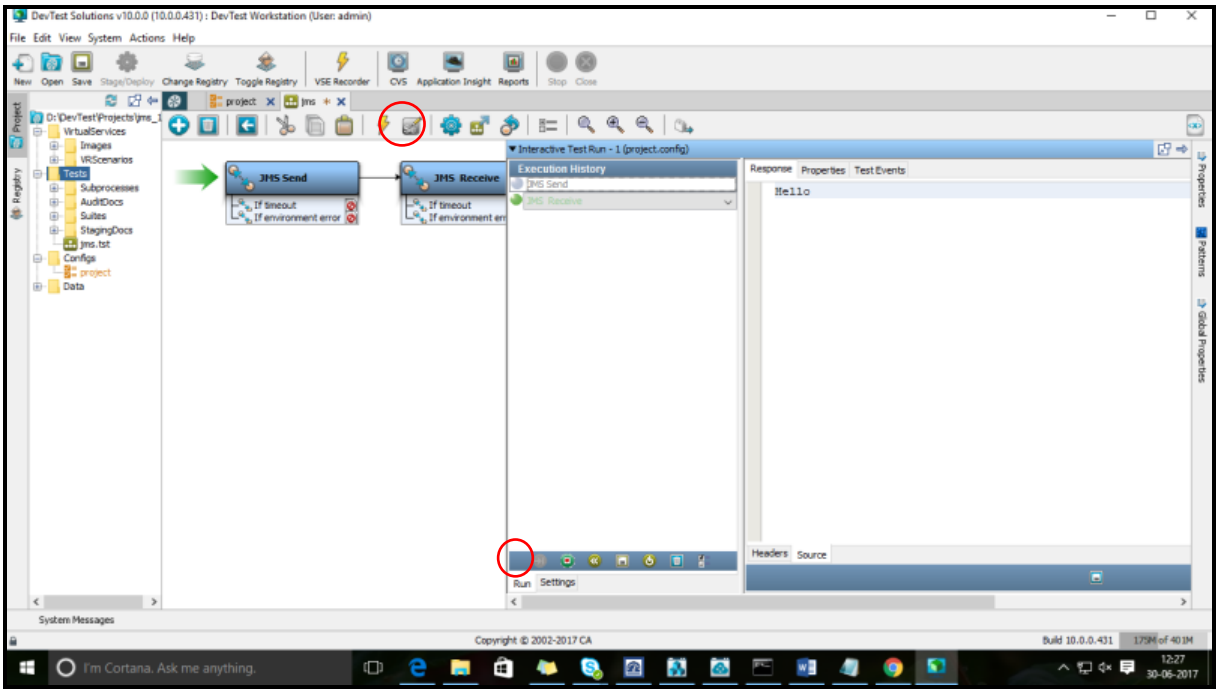
Uncheck JMS Receive.



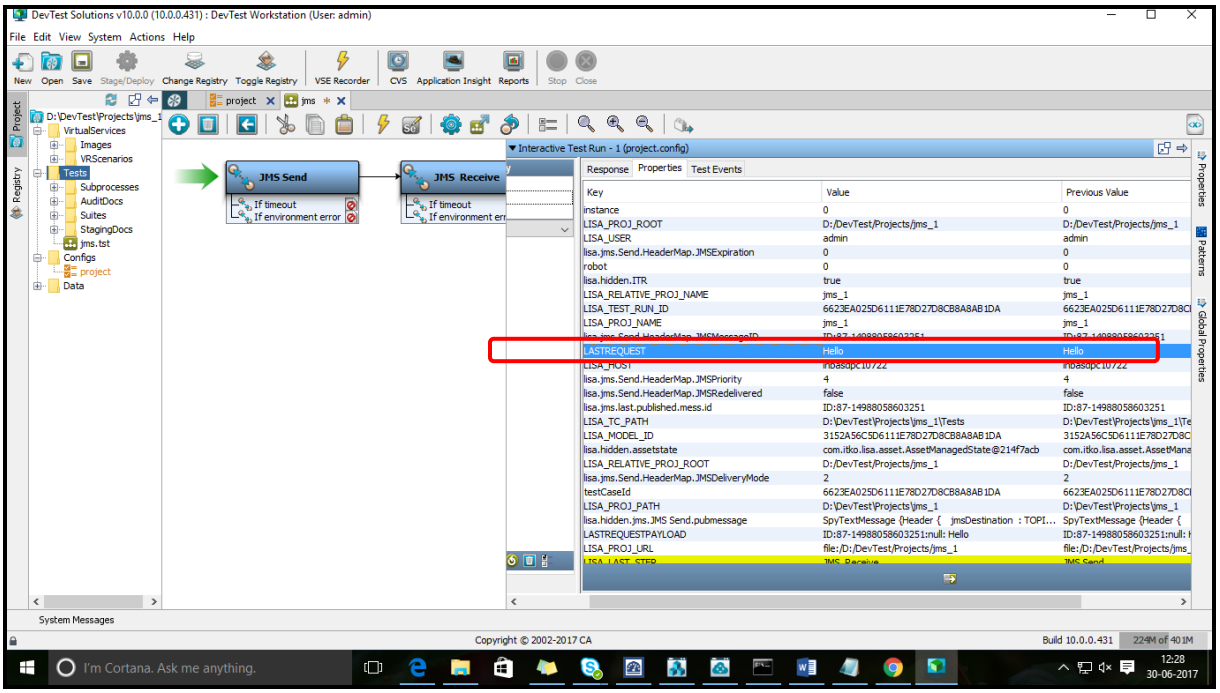
6. Add another JMS Messaging Step > JMS Send Receive. Uncheck JMS Send . Click on  .
   Message will be displayed in content area.



7. Start ITR and click automatically run.

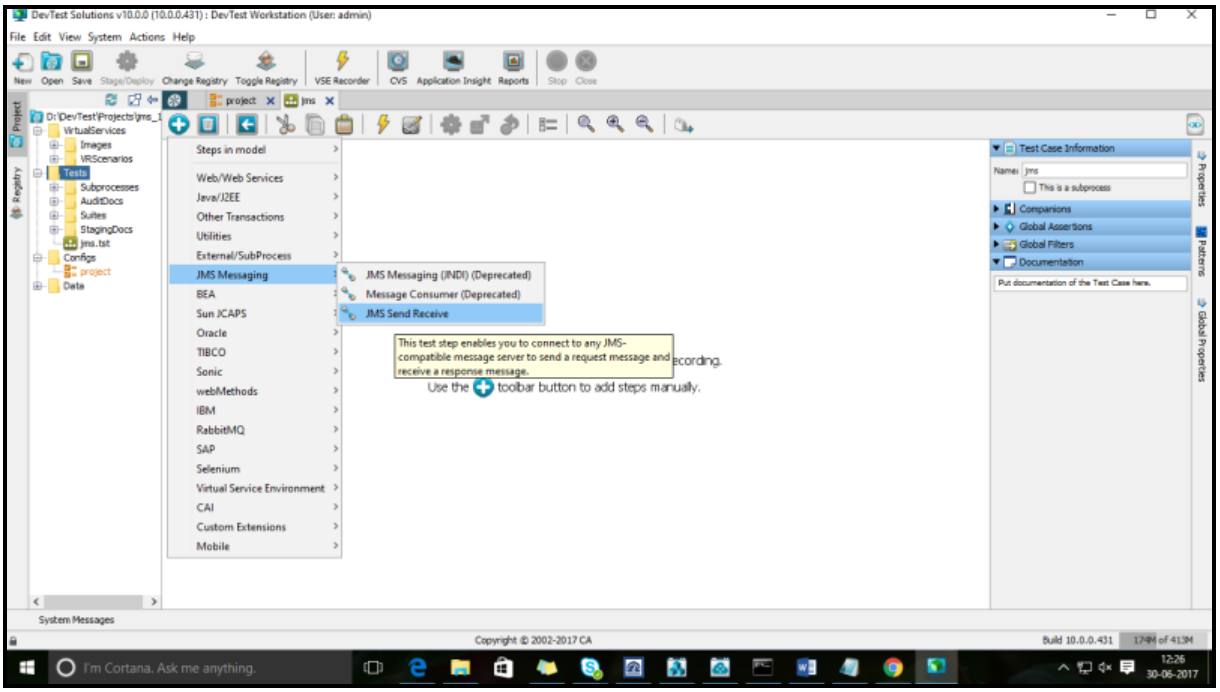


8. We can see the message in properties tab > LastRequest.

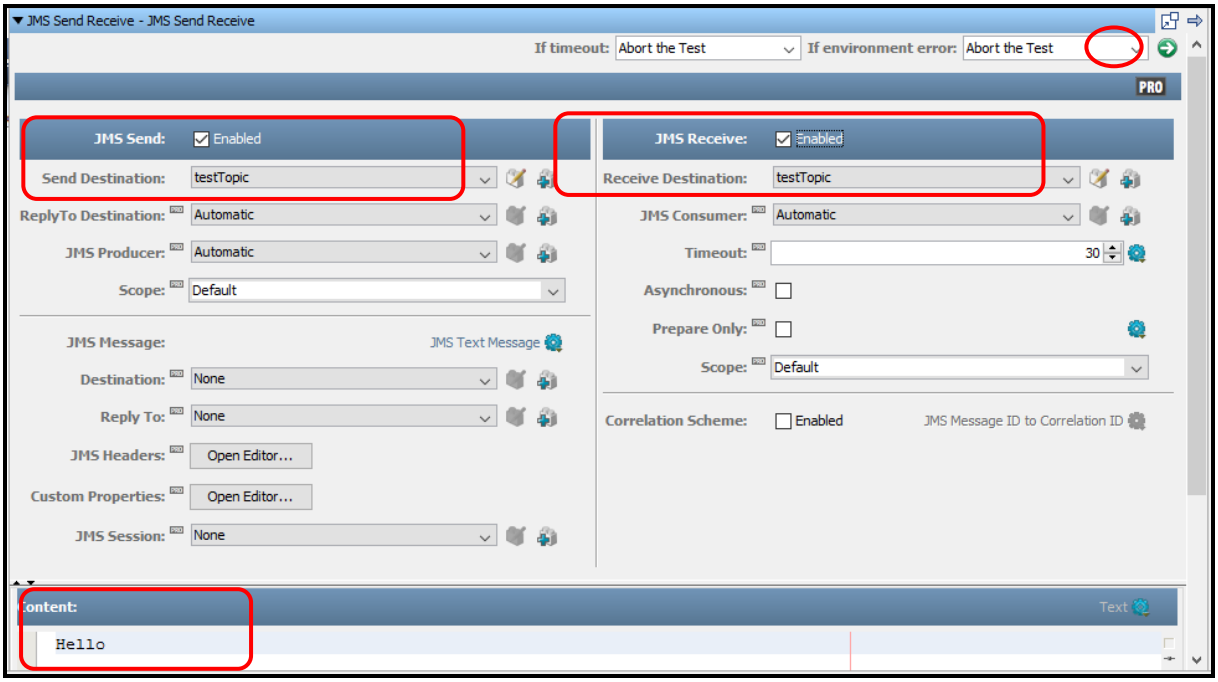


<u>Steps for sending and receiving in a single step.</u>

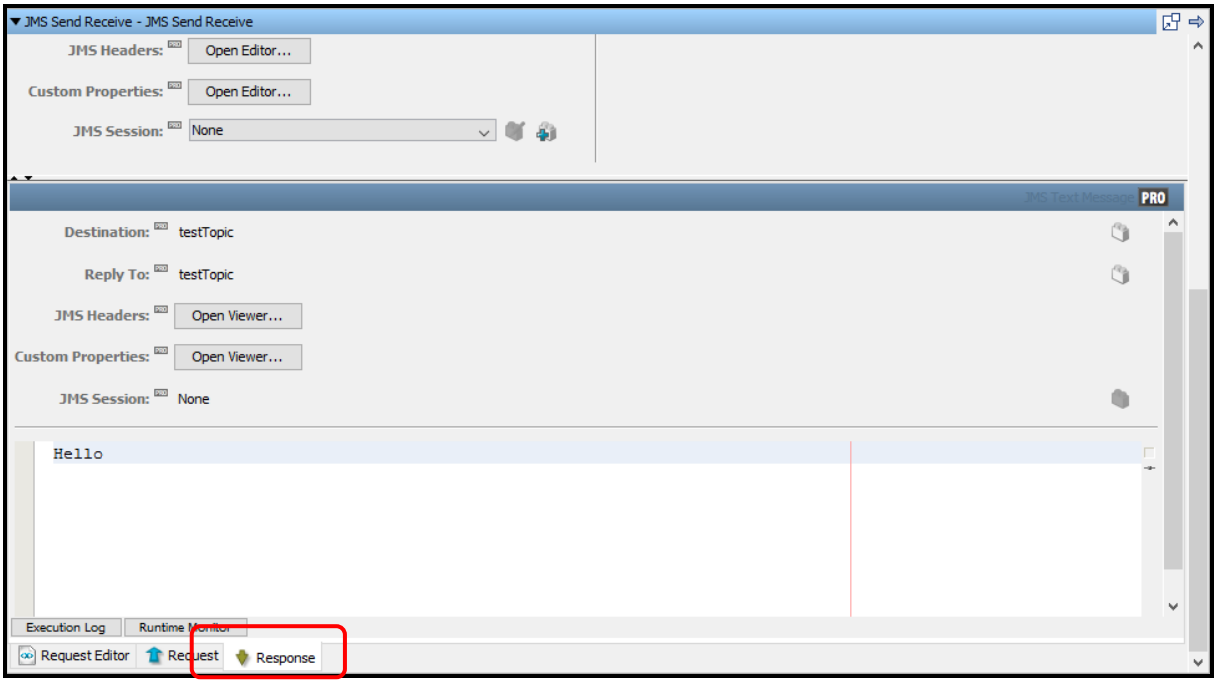
1. Create new Test Case >ADD JMS Messaging Step > JMS Send Receive



2. Select the topic created earlier in Send destination and Receive destination. Write in Content area. Check both JMS send and JMS Receive. Click on  .



3. Message is received in response.



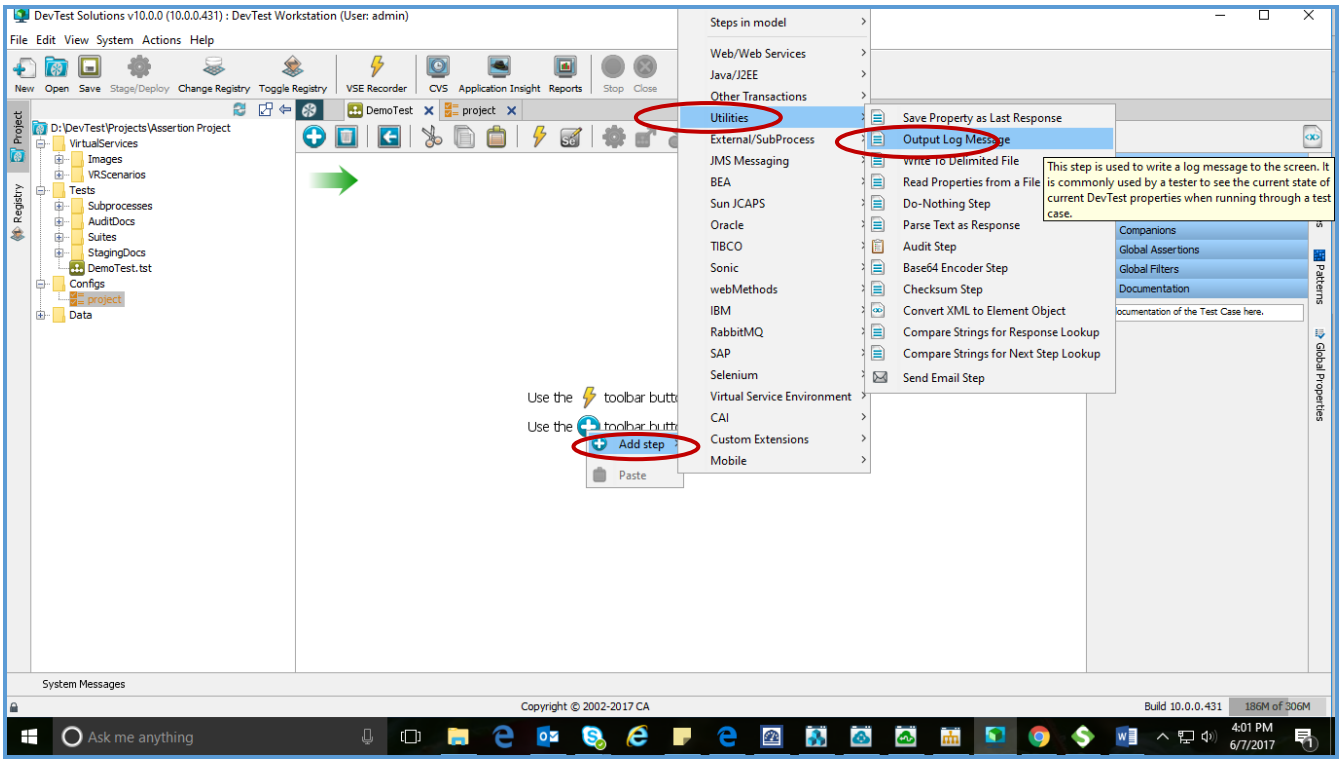
18. Filters

The filters that the DevTest Solutions provides includes most of the logic that is required to test enterprise software. However, you can create your own filter to handle a specific situation. DevTest provides built-in support for custom filters.

You can add various types of filters to a test case. In this document **Utility Filter** with the name **Store Step Response** is demonstrated. This type of filter lets you **save** the **step response** as a **property**.

18.1 How to add Filters

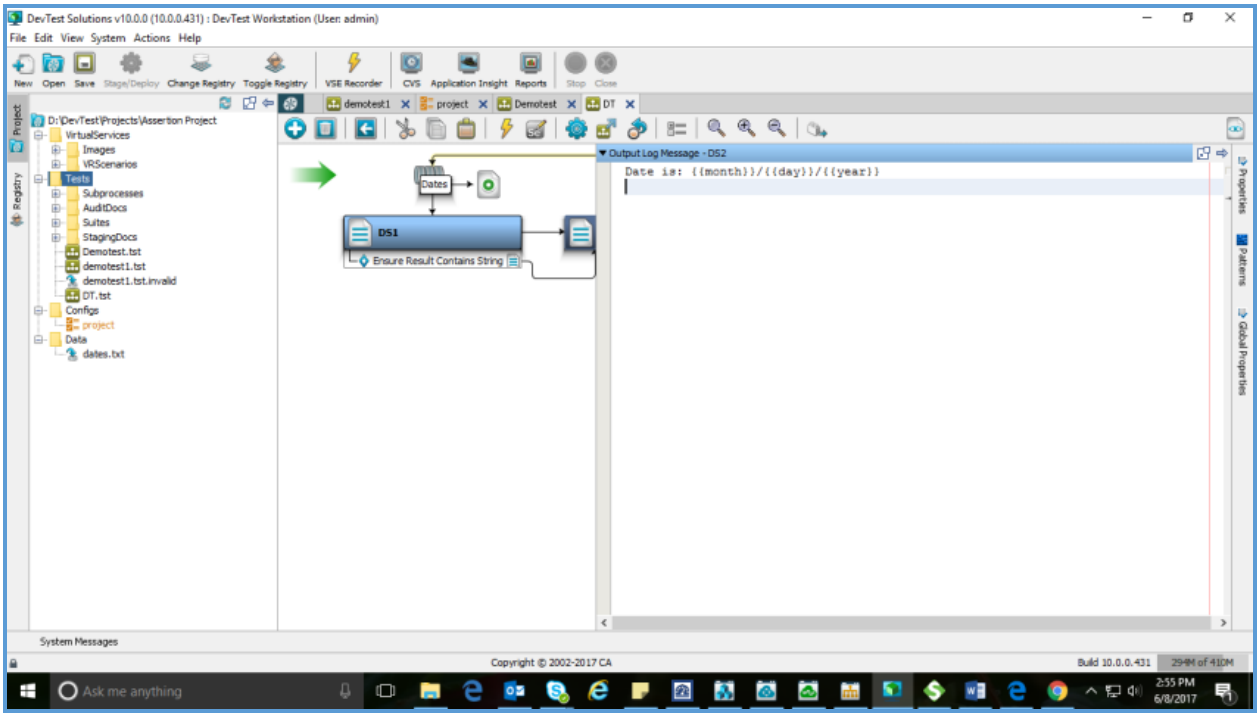
1. Select a Test case from project panel or right-click on Test folder and create a new test case. A new test case is created in this example
2. Add two steps to the test case. Add **Output Log Message** from **Utilities** and rename by right-click. DS1 & DS2 are names used in this example



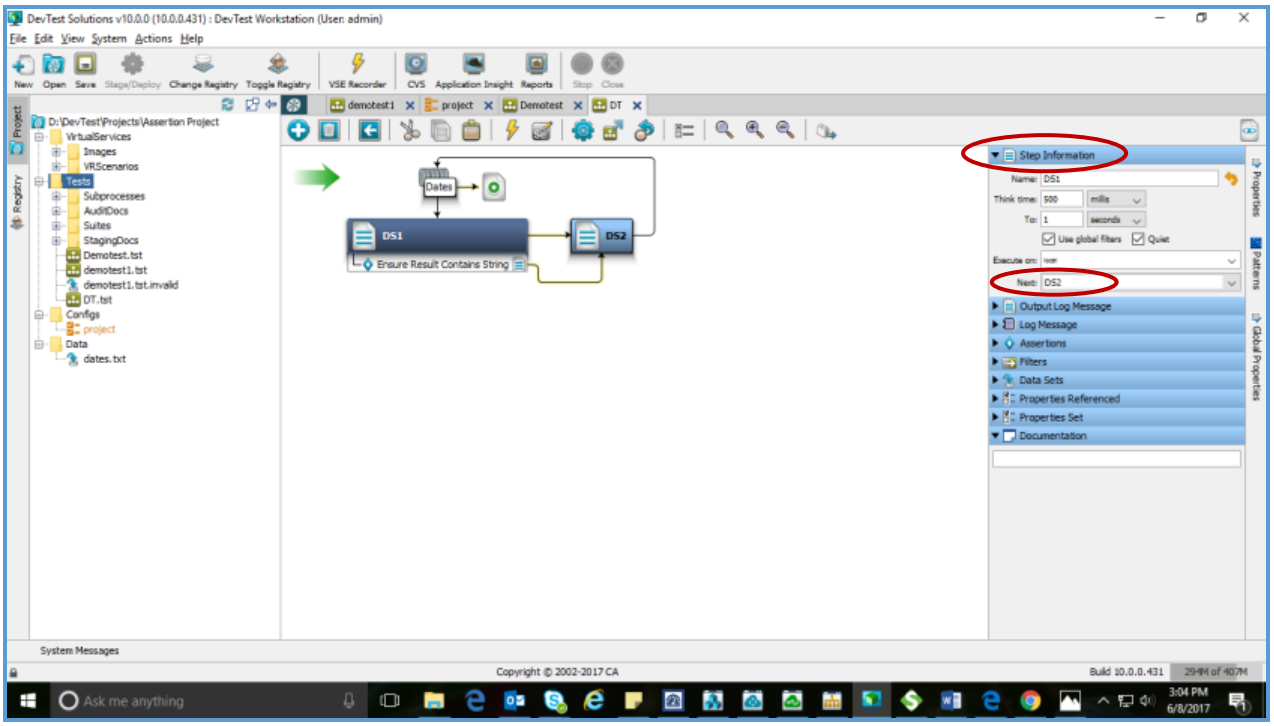
3. Double-click on DS1 in the model editor and change the Output Log Message to

```
Date is: {{month}}/{{day}}/{{year}}
```

4. Repeat step 3 for DS2



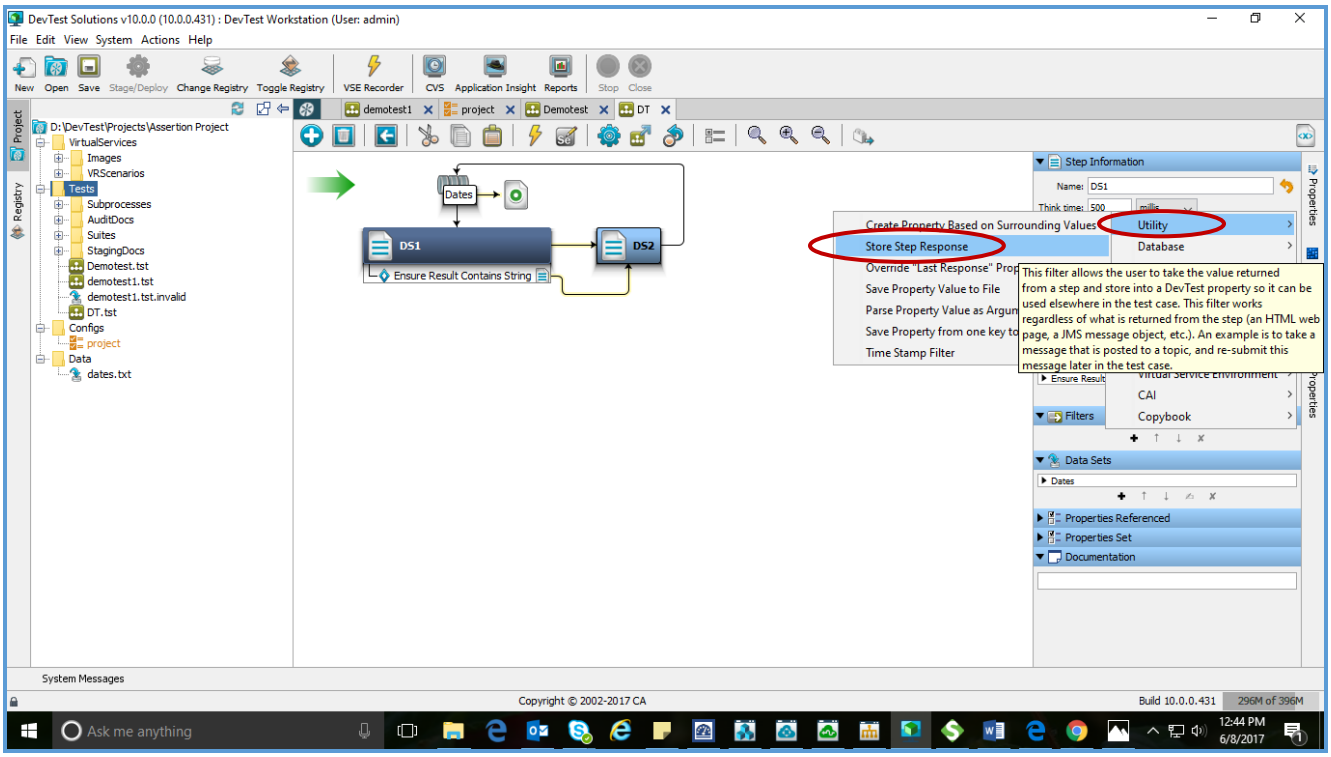
5. In the model editor select DS1 and on the right pane, under **Step Information** select **DS2** from dropdown list under **Next** and vice-versa with DS2



6. In the model editor select **DS1** and on right pane add data text file in datasets

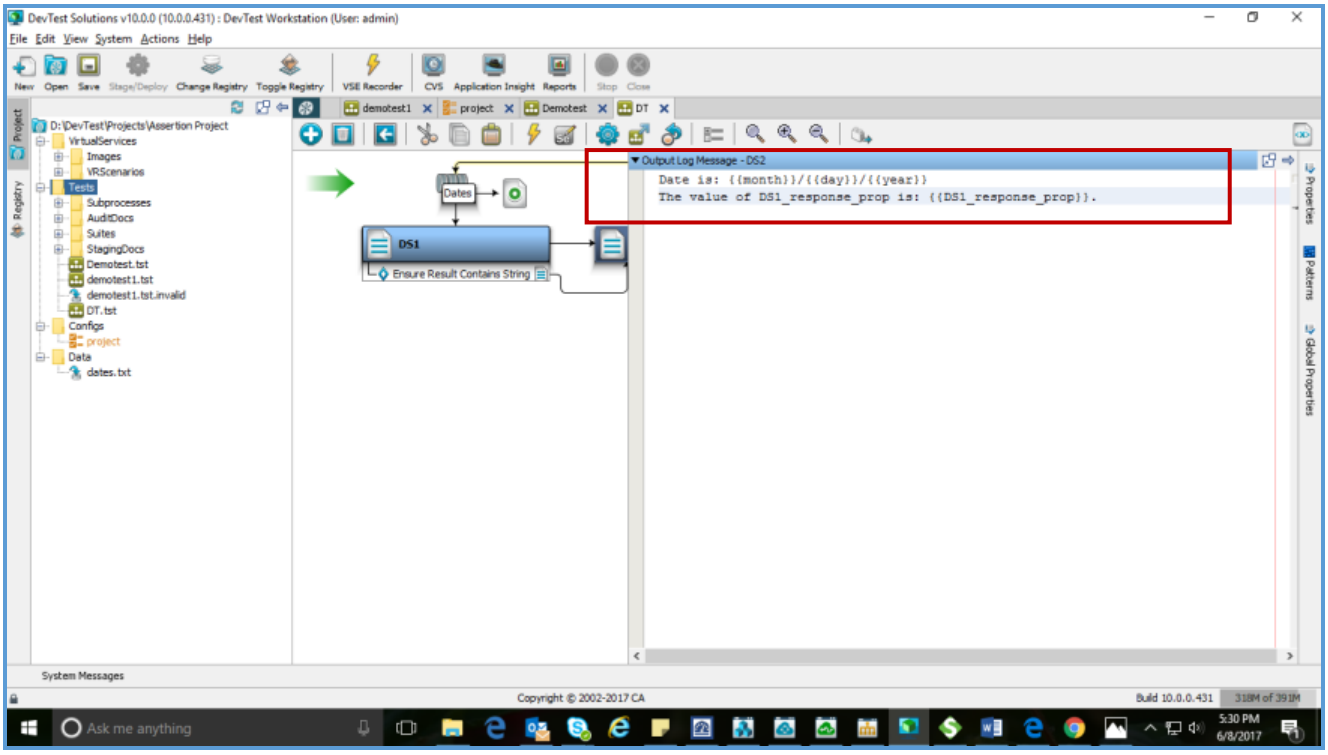
   Refer: Working with datasets: https://docops.ca.com/devtest-solutions/8-0/en/getting-started/ca-application-test-tutorials/tutorial-2-data-sets

7. In the model editor select **DS1,** Open the Assertions tab on the right pane. Click **Add**. From the **Utility Filters** submenu, select **Store Step Response**. The filter editor opens

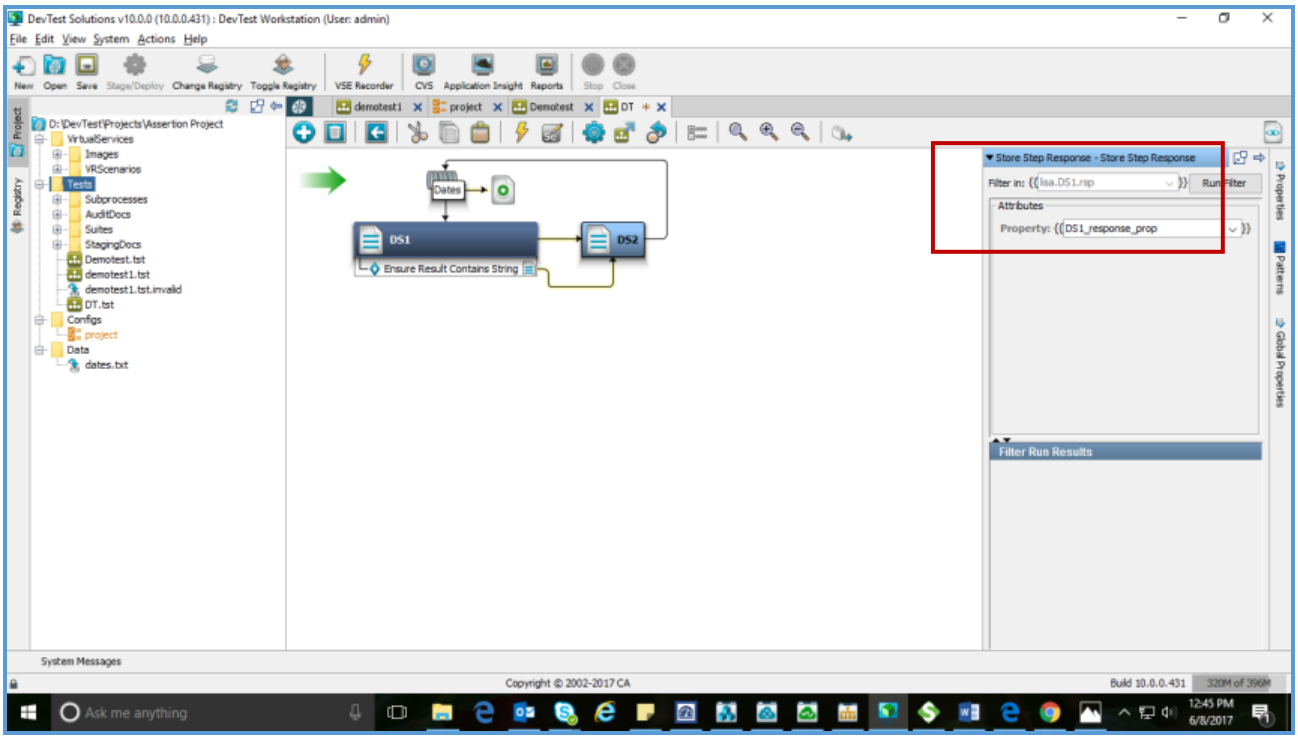


8. In the model editor, double-click **DSstep2** and add the following text to the end of the output log message:

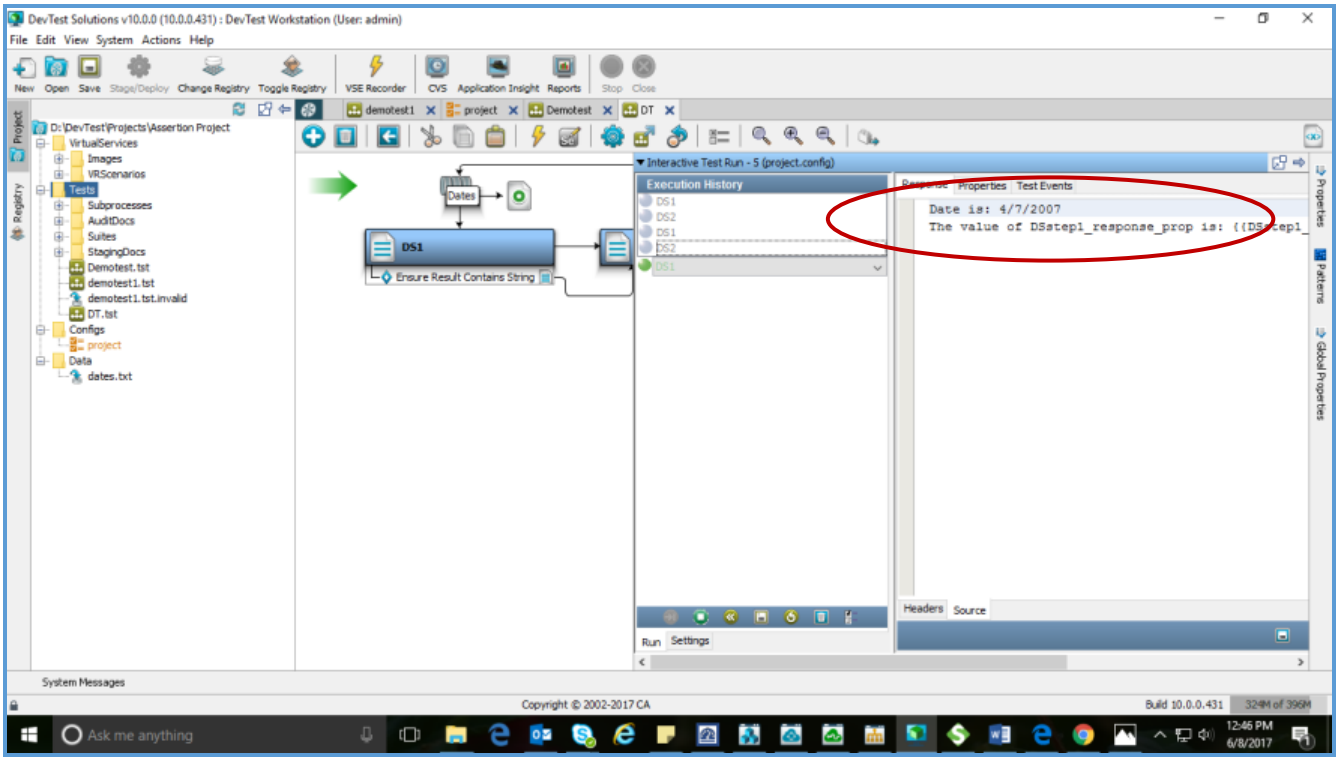
```
The value of DSstep1_response_prop is: {{DSstep1_response_prop}}
```



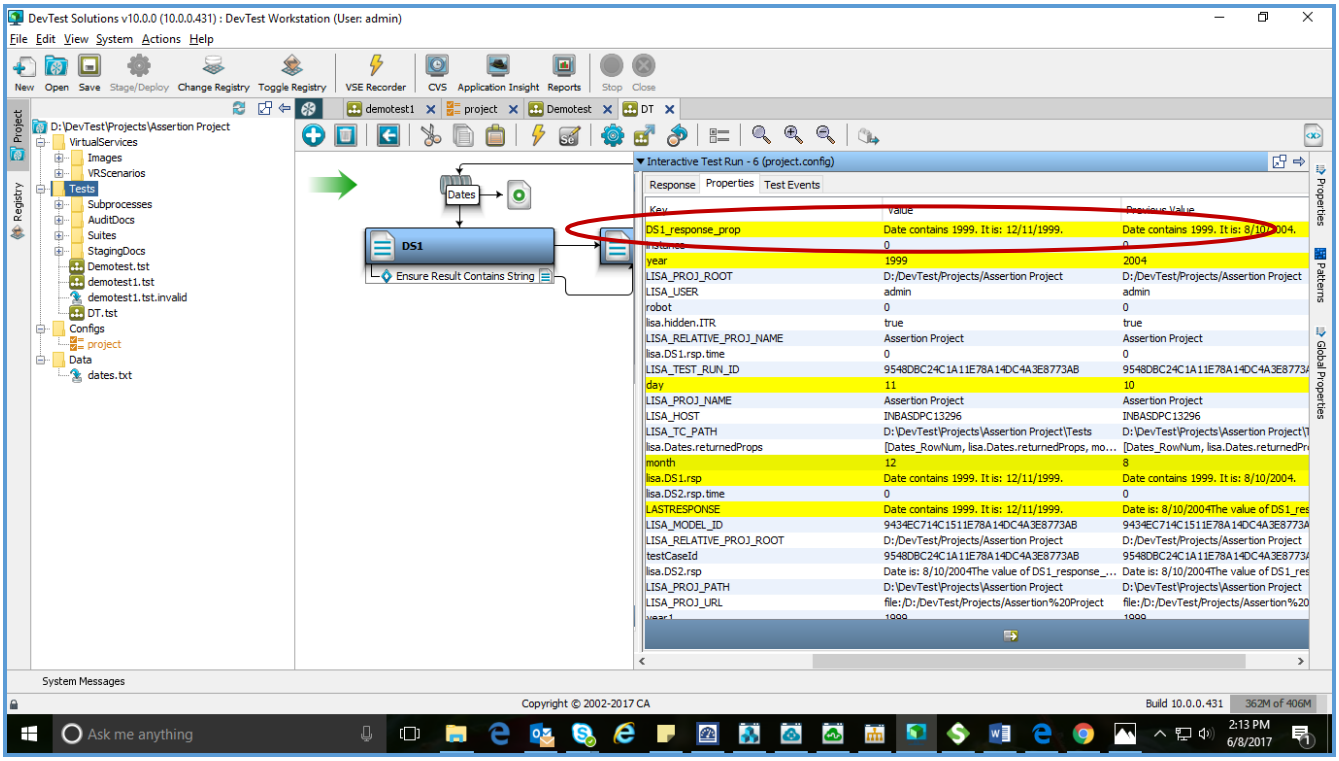
9. In the filter editor, set the property name to **DS1_response_prop** and save. This property is where the step response is stored



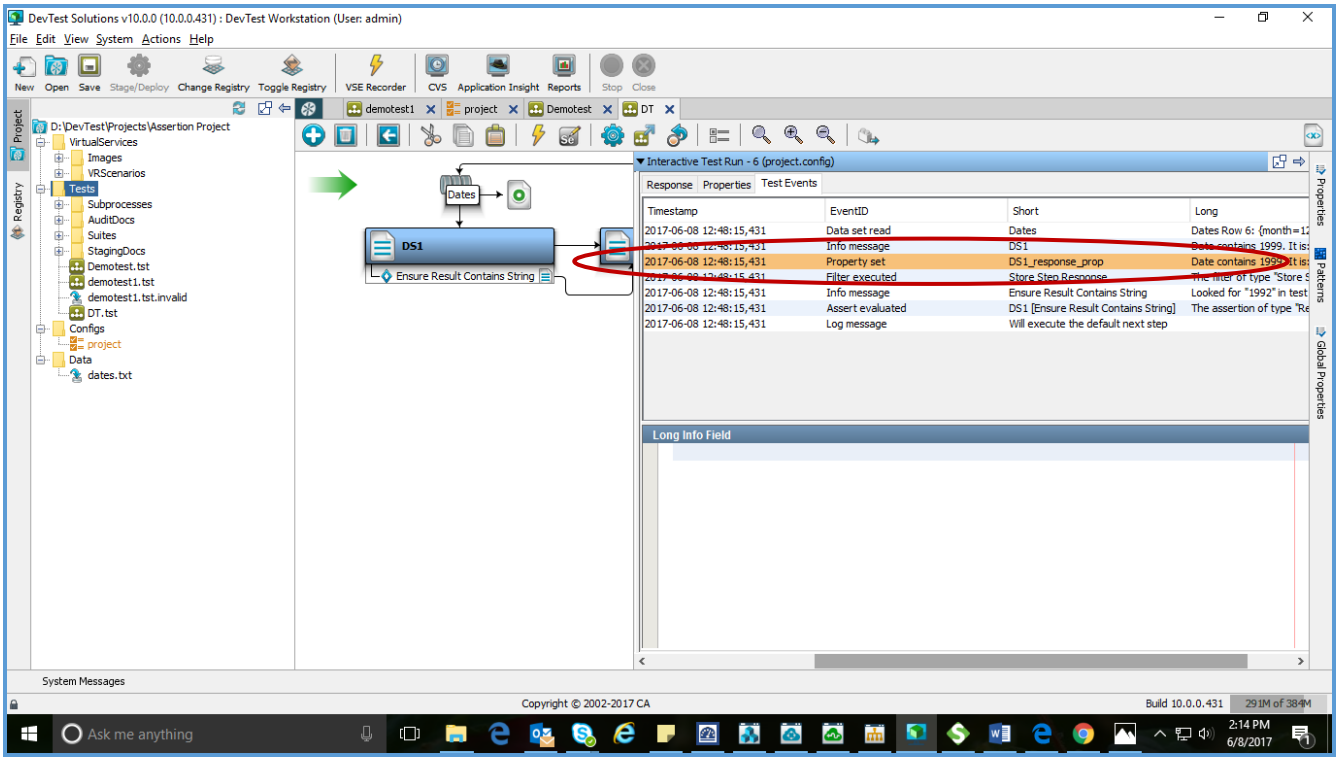
10. Start a ITR session and review the response time.



11. Click the **Properties tab** and observe where the **DS1_response_prop** property is created and modified



12. Click the **Test Events** tab and **review** the events that were generated.



Note:

To explore more on filters, go through the following link.

https://docops.ca.com/devtest-solutions/8-0-1/en/reference/test-case-reference/filter-descriptions

19. Assertions

An **assertion** is a DevTest code element that runs after a step and all its filters have run. An assertion verifies that the results from the step match expectations. The result of an assertion is a Boolean value (true or false). The outcome can determine whether the test step passes or fails, and also determines the next step to run in the test case. An assertion is used to change the test case workflow dynamically by introducing conditional logic (branching) into the workflow. An assertion operates very much like an "if" conditional block in programming.

Manual assertions are of two types:

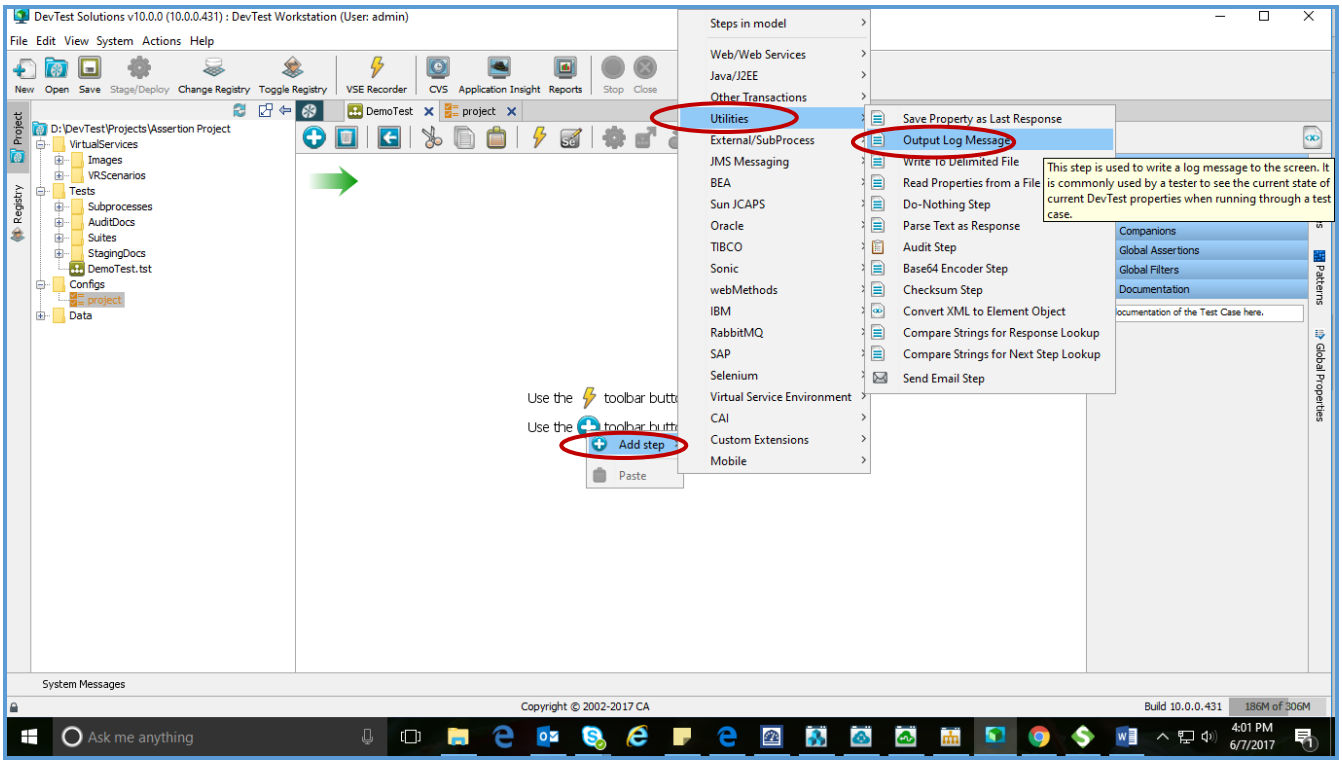
1. **Global assertions:** Global assertion are defined at the test case level. A global assertion applies to all the steps in the test case and is automatically run for every step, unless a node is instructed otherwise
2. **Step assertions:** Step assertions are defined at the test step level. This type of assertion applies only to that step and executes for that step only

19.1 How to add Assertion

1. Select a Test case from project panel or right-click on Test folder and create a new test case. A new test case is created along with data sets in this example

2. Add two steps to the test case. Add **Output Log Message** from **Utilities** and rename by right-click.

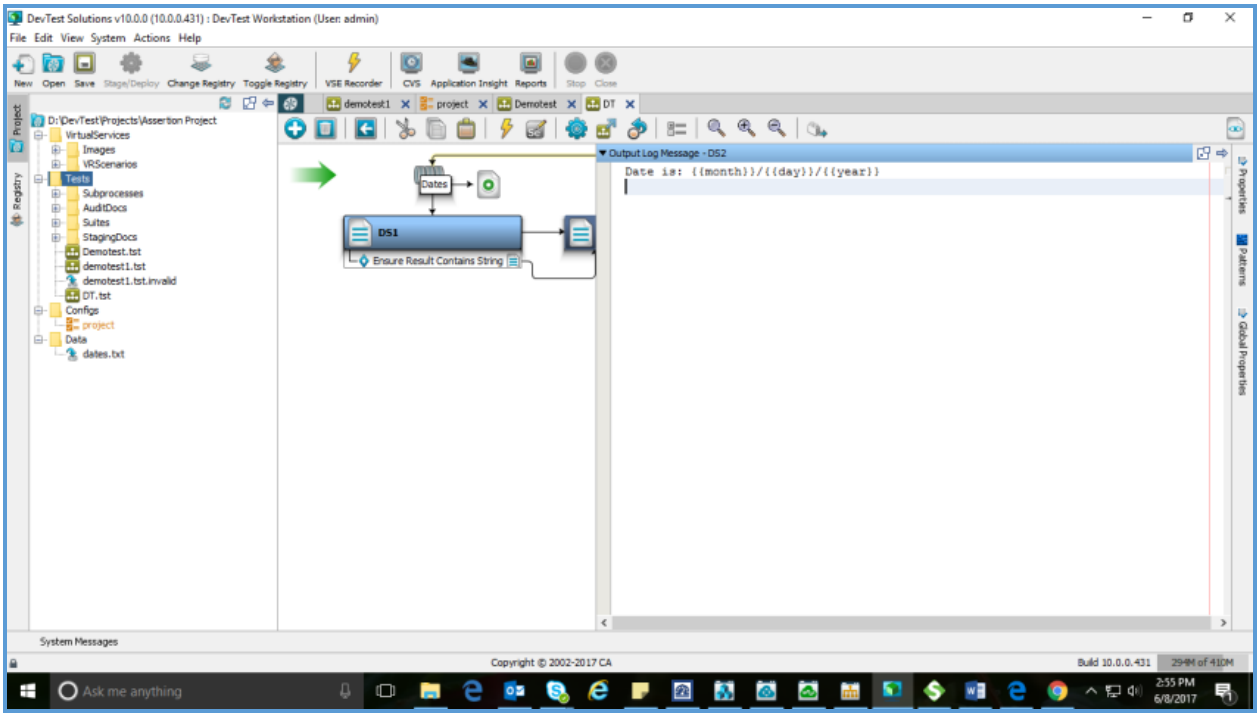
   DS1 & DS2 are names used in this example



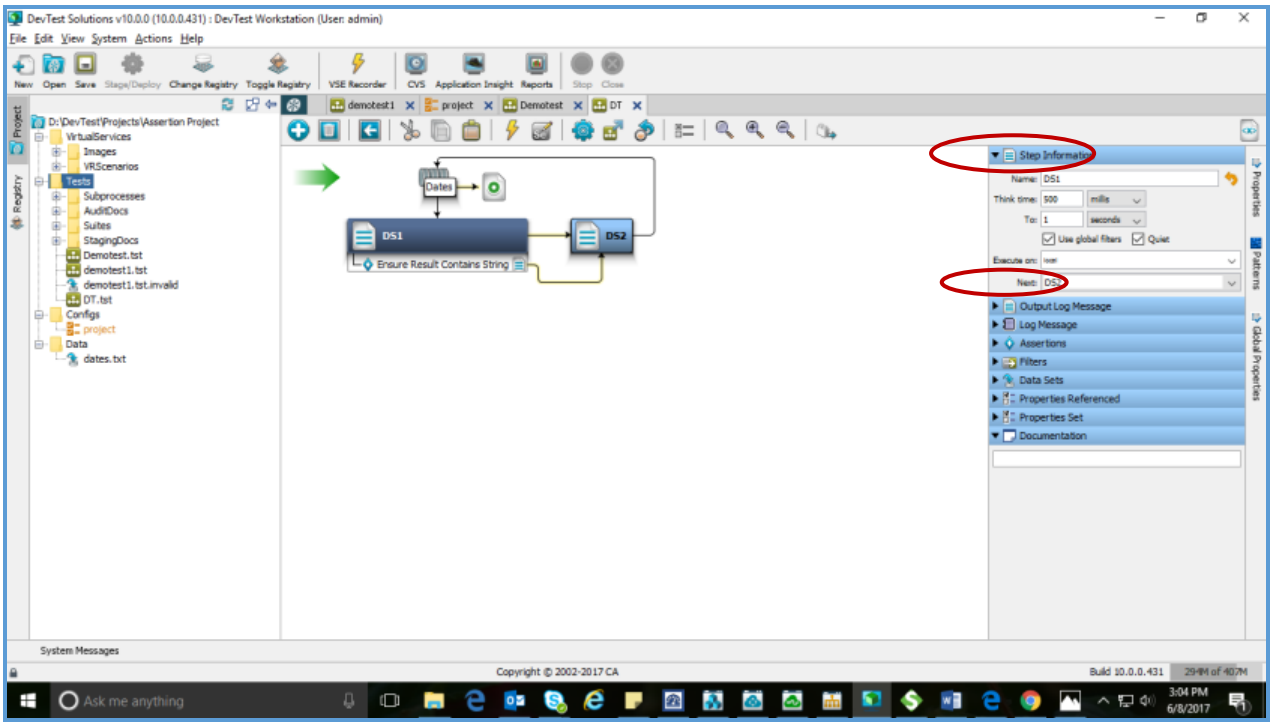
3. Double-click on DS1 in the model editor and change the Output Log Message to

```
Date is: {{month}}/{{day}}/{{year}}
```

4. Repeat step 3 for DS2



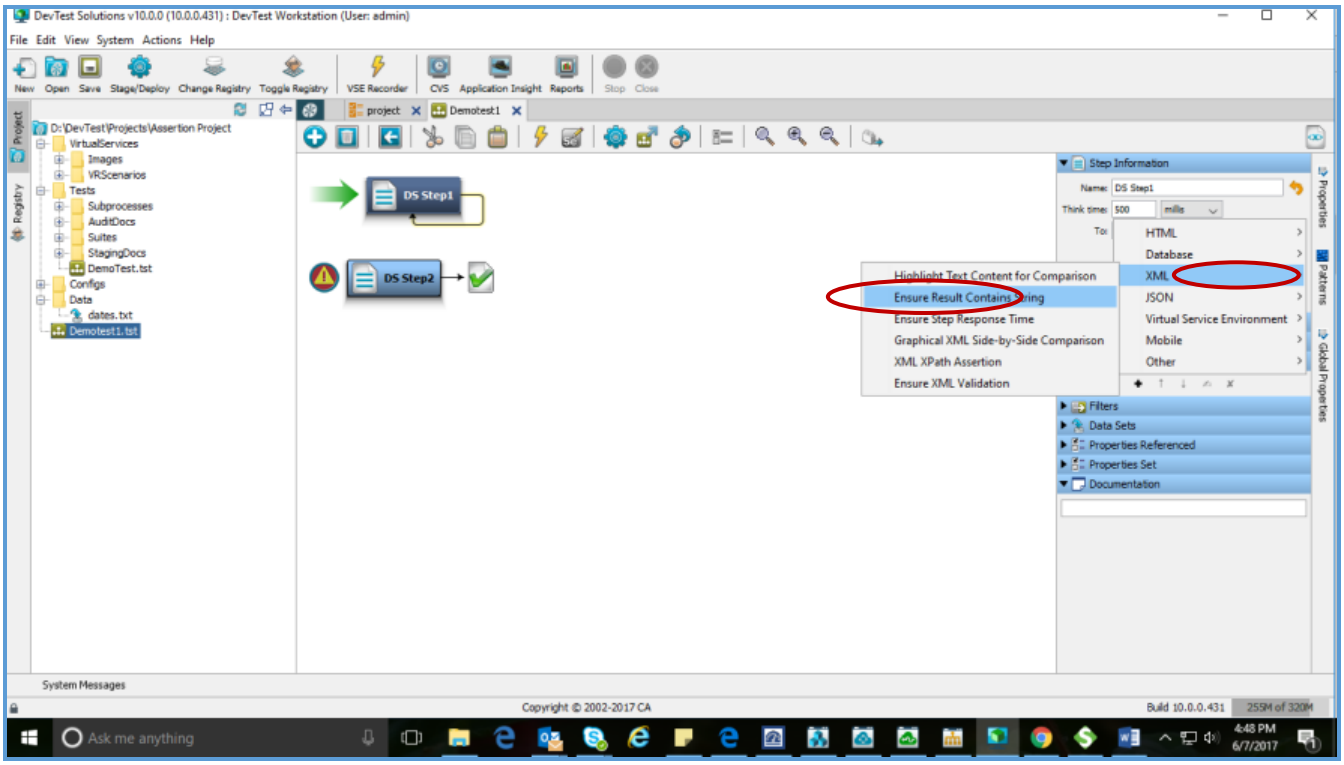
5. In the model editor select DS1 and on the right pane, under **Step Information** select **DS2** from dropdown list under **Next** and vice-versa with DS2



6. In the model editor select **DS1** and on right pane add data text file in datasets.

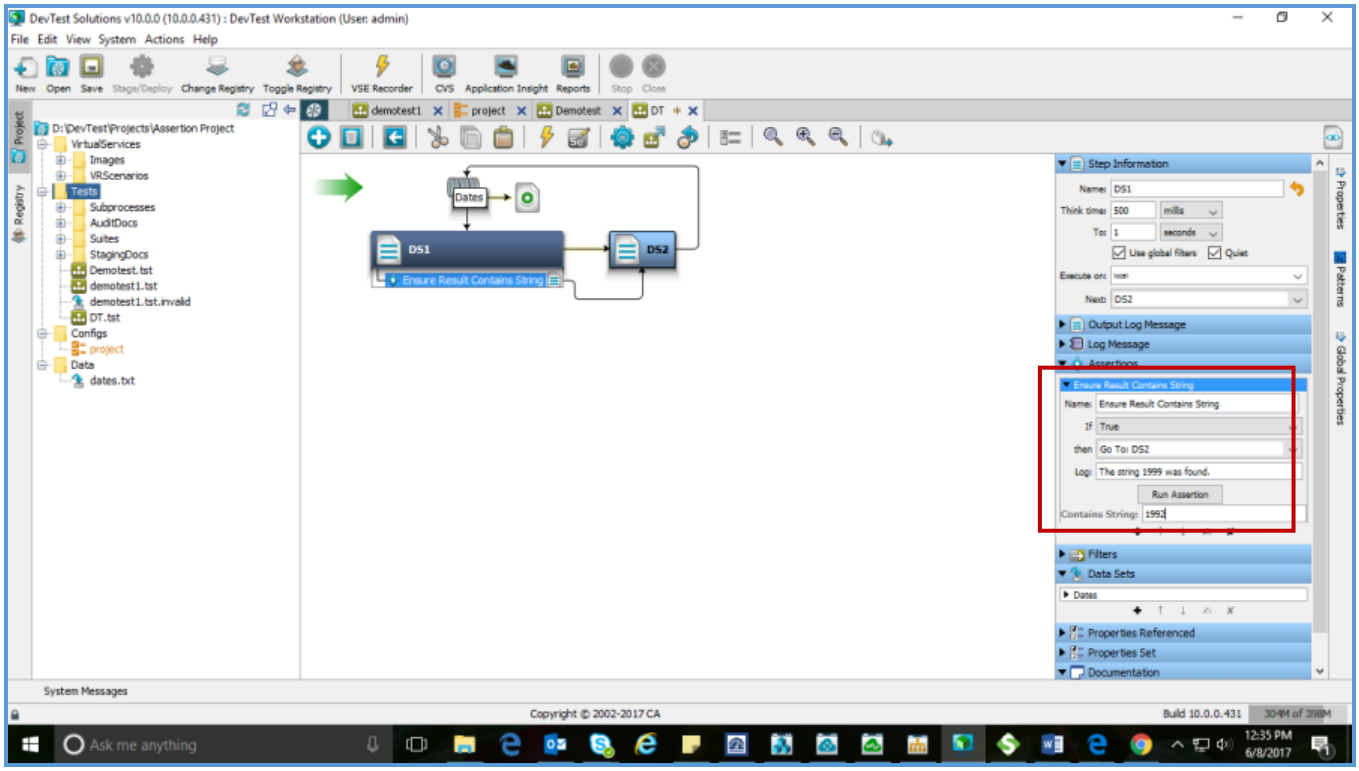
   Working with datasets: https://docops.ca.com/devtest-solutions/8-0/en/getting-started/ca-application-test-tutorials/tutorial-2-data-sets

7. In the model editor select **DS1,** Open the **Assertions** tab on the right pane. Click **Add**. From the XML submenu, select **Ensure Result Contains String**

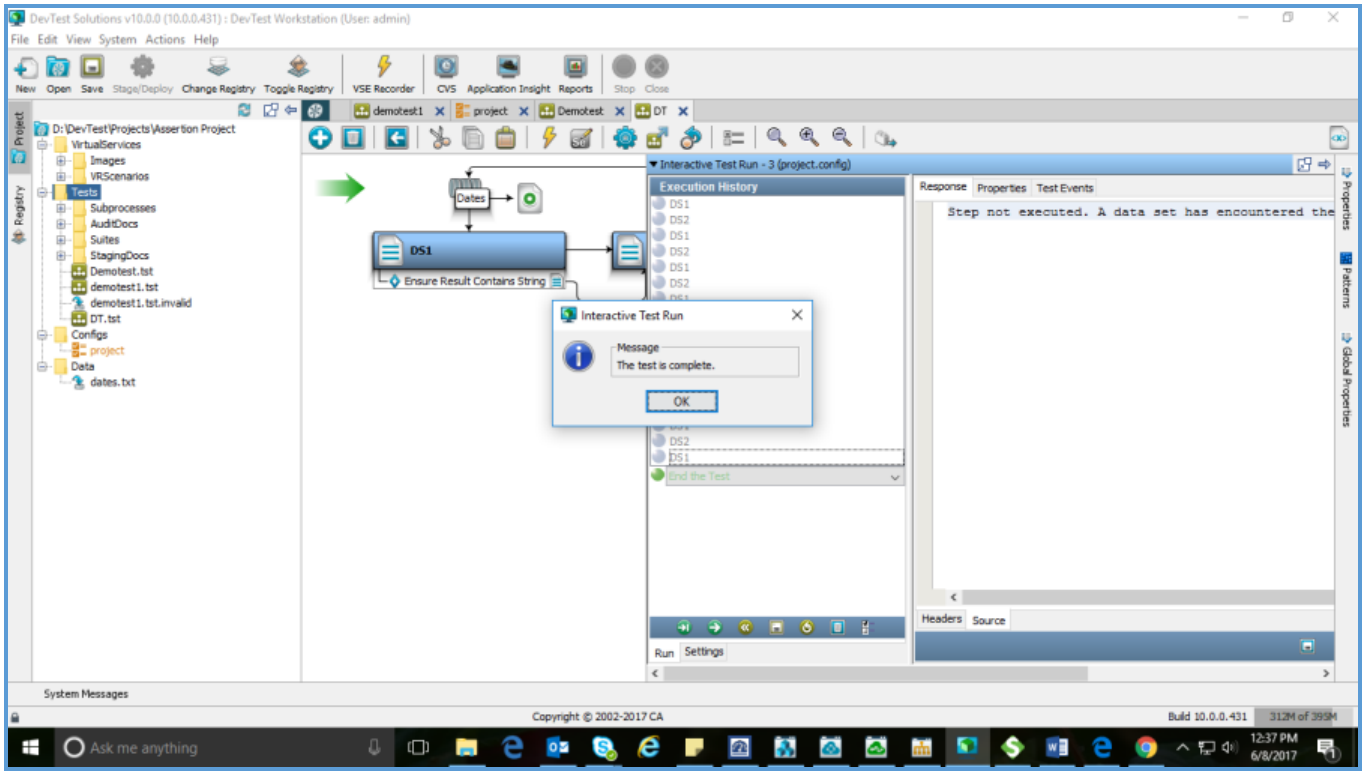


The new assertion that is applied to DS1 is added to the Assertions tab.

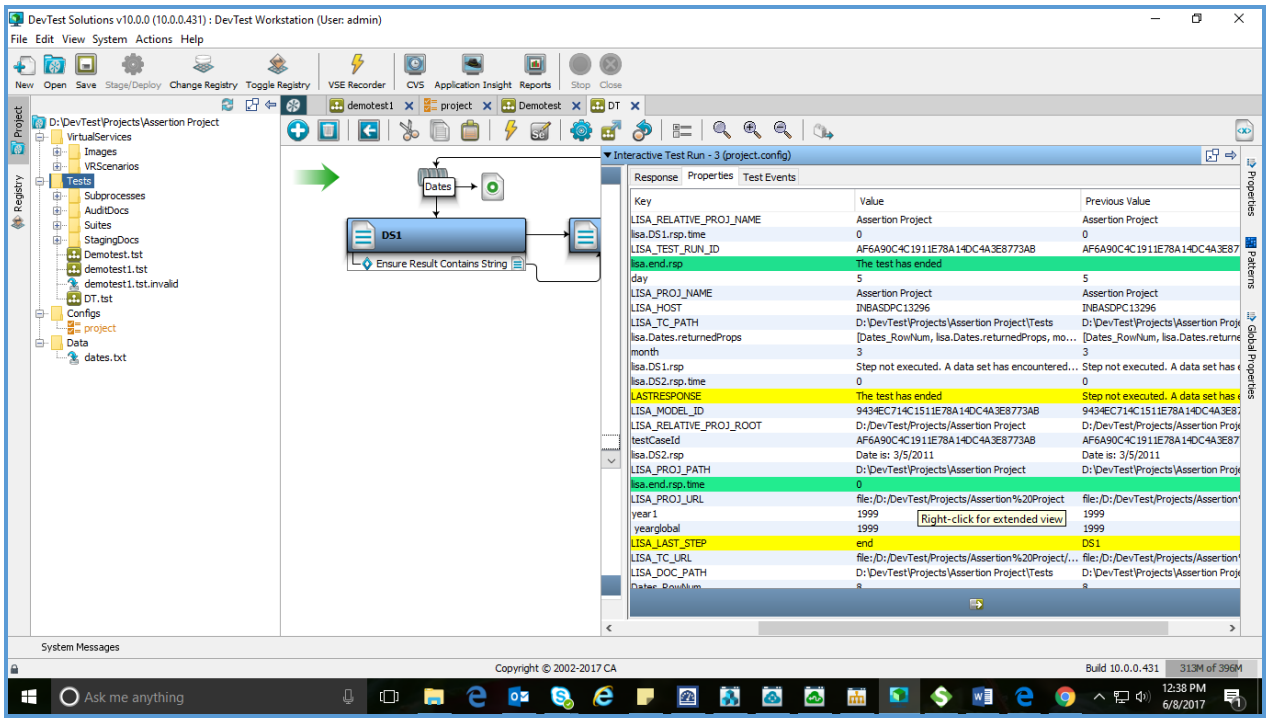
8. The assertion editor opens. In the assertion editor, do the following:
    a. In the **If list**, select **True**
    b. In the **then list**, select Go To: **DS2**
    c. In the **Log field**, enter **The string 1999 was found**
    d. In the **Contains String field**, enter **1999** and save



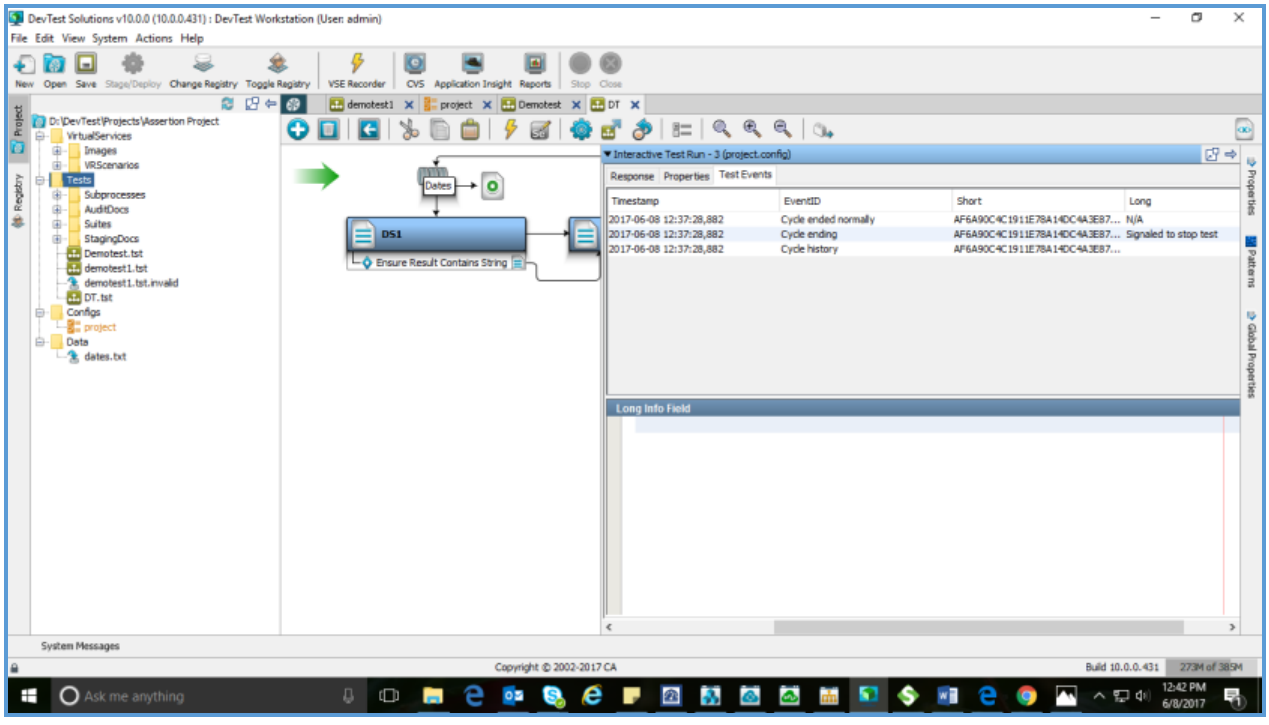
9. Start a ITR session



10. Click the **Properties tab** and **review** the behavior of the properties.



11. Click the **Test Events tab** and review the events that were **generated**



Note: To explore more on assertions, go through the following link.

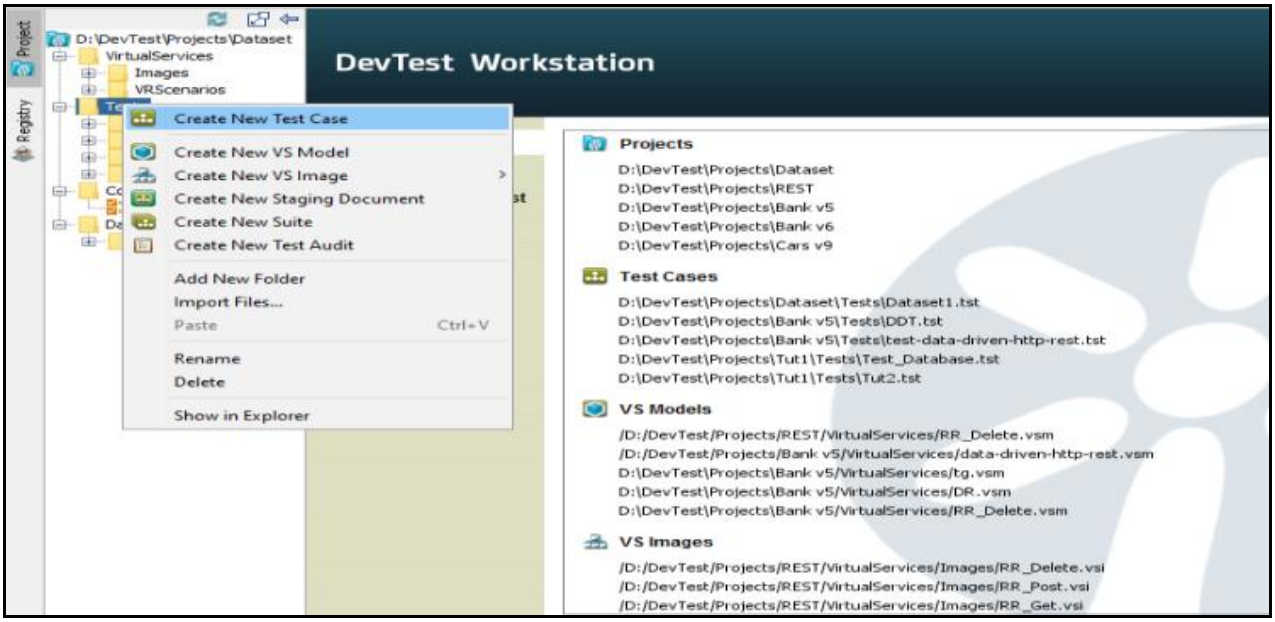
https://docops.ca.com/devtest-solutions/8-0/en/reference/test-case-reference/assertion-descriptions

20. Datasets

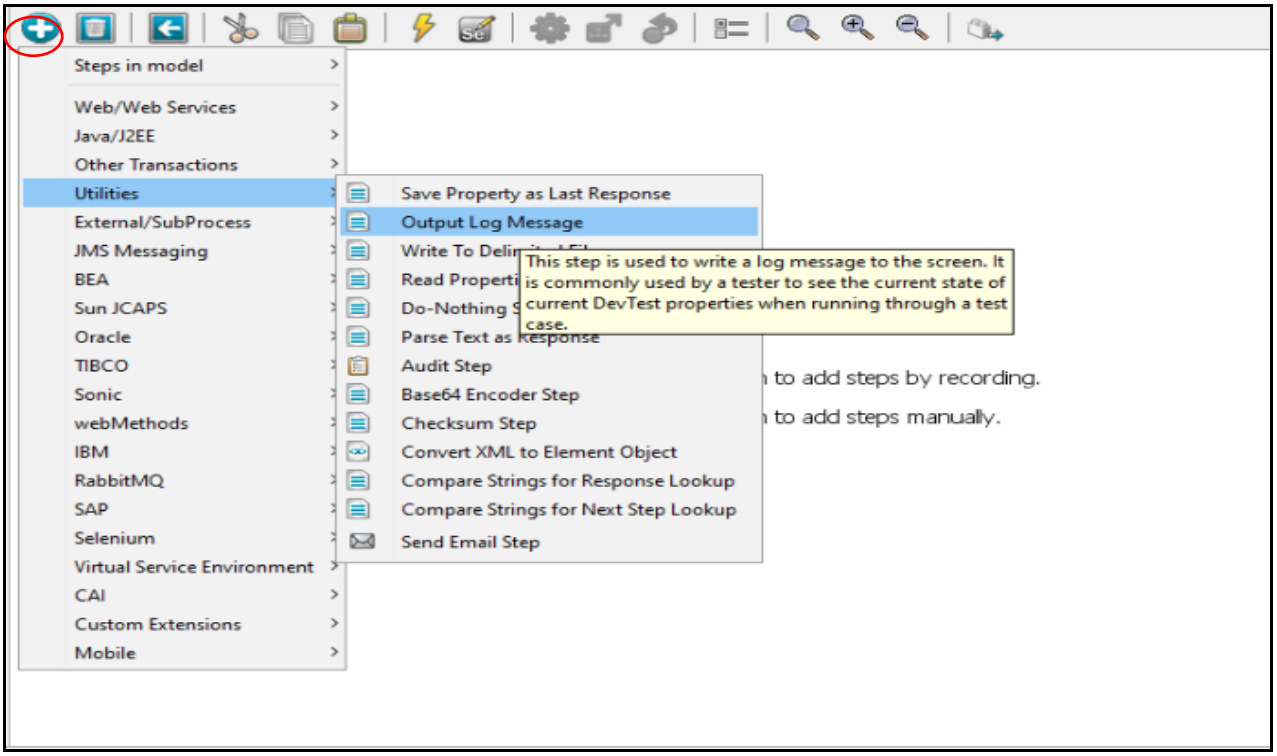
A data set is a collection of data. A data set corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the data set.

Below are the steps for creating a simple data set and iterating through a series of test step using that data set.

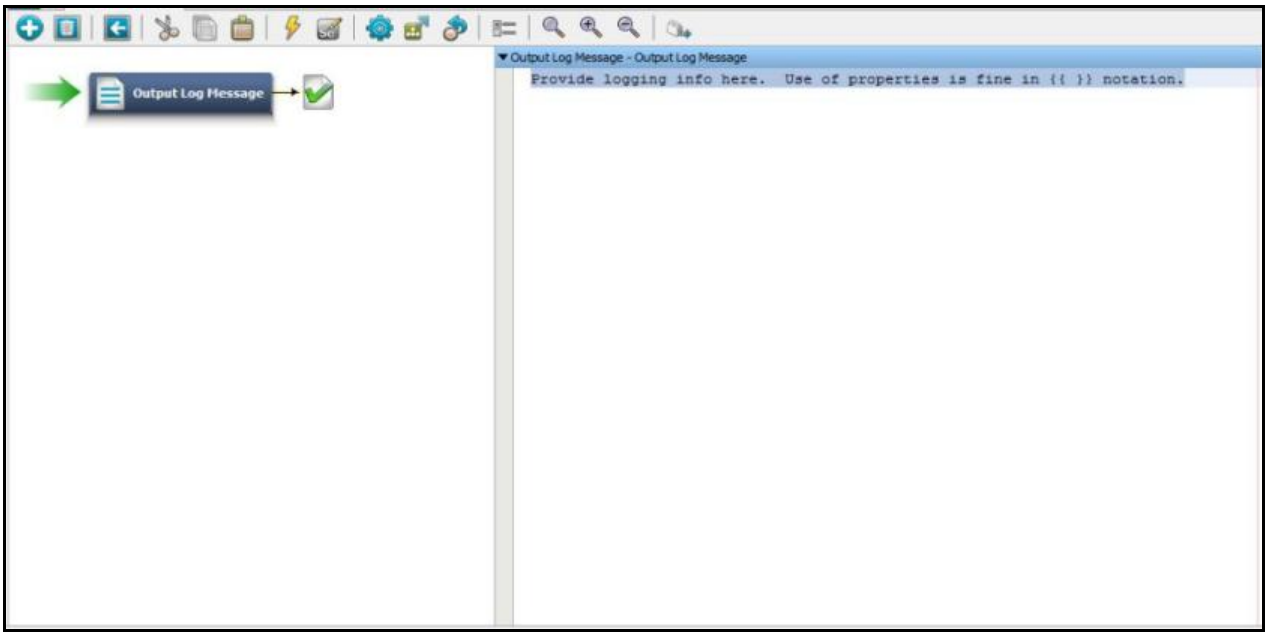
- Project > Tests > Create New Test Case.



- Click Add. Select Utilities and select Output Log Message.

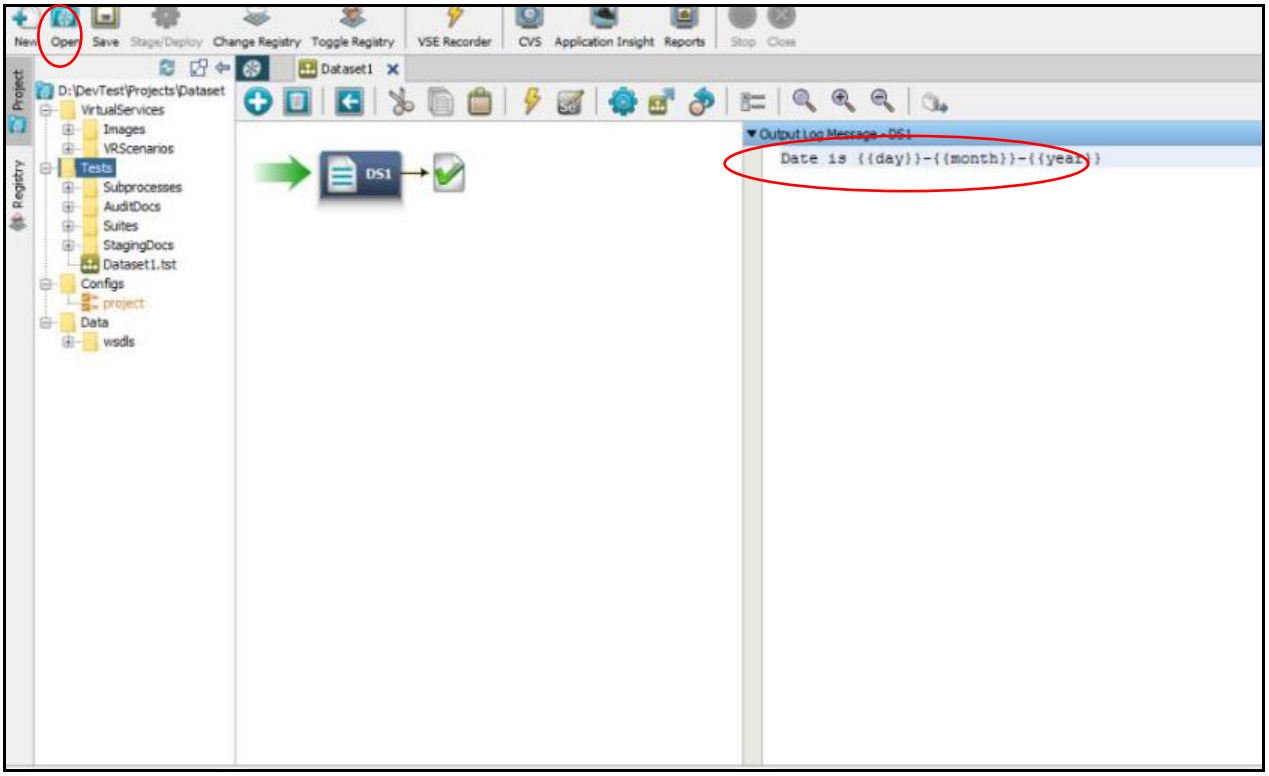


- A step with the name **Output Log Message** is added to the model editor.

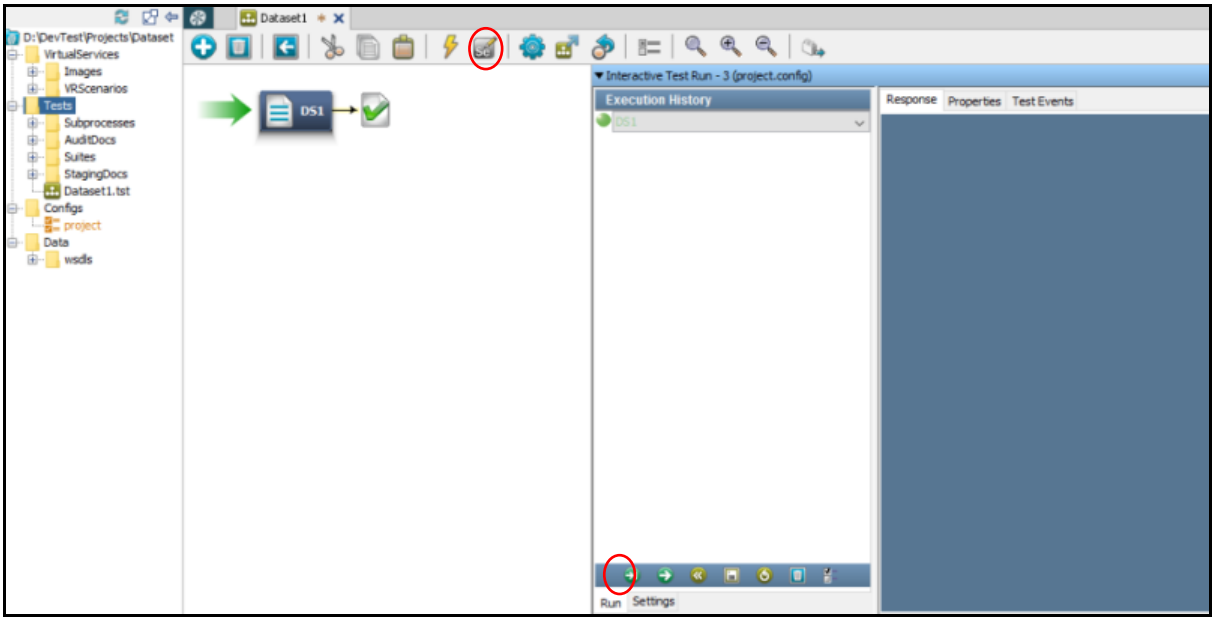


- Right-click Output Log Message and select Rename. Change the name to DSstep1.

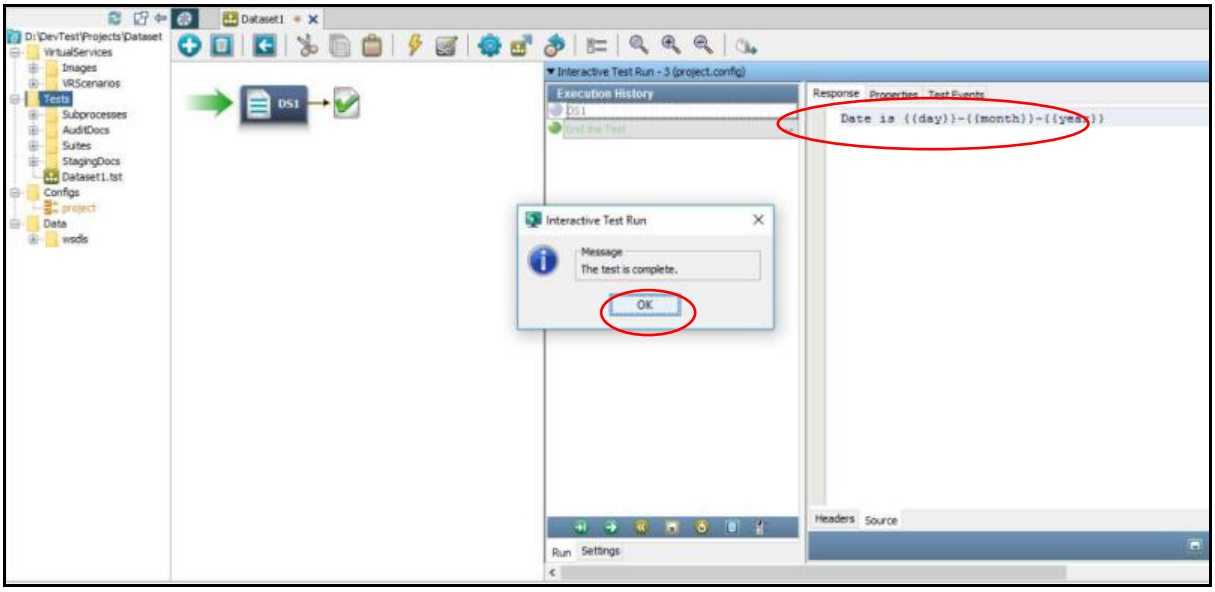
In the Output Log Message tray enter: Date is {{day}}-{{month}}-{{year}}. Click Save.



- Start ITR and click Automatically execute test.

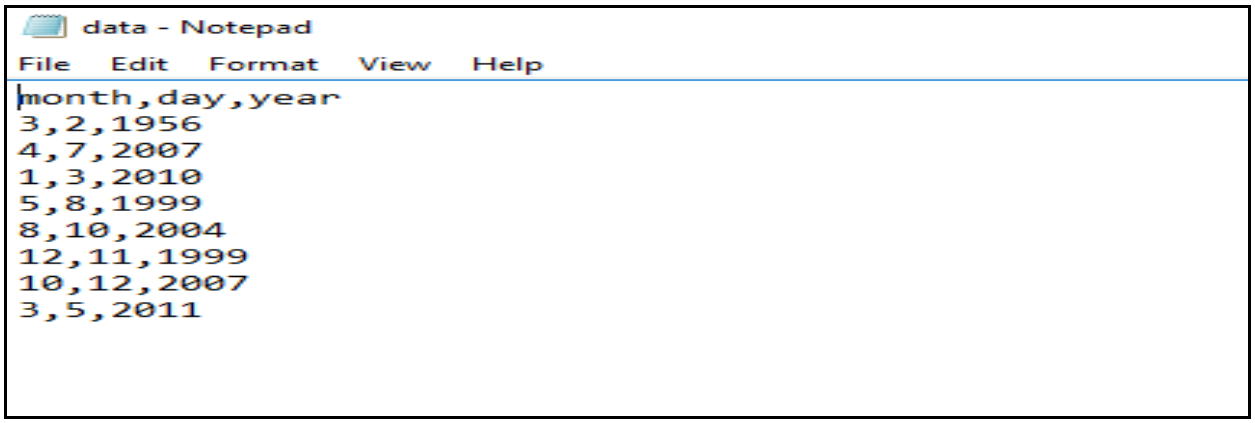


- When the test is complete, click OK. In the Execution History pane, click DSstep1.

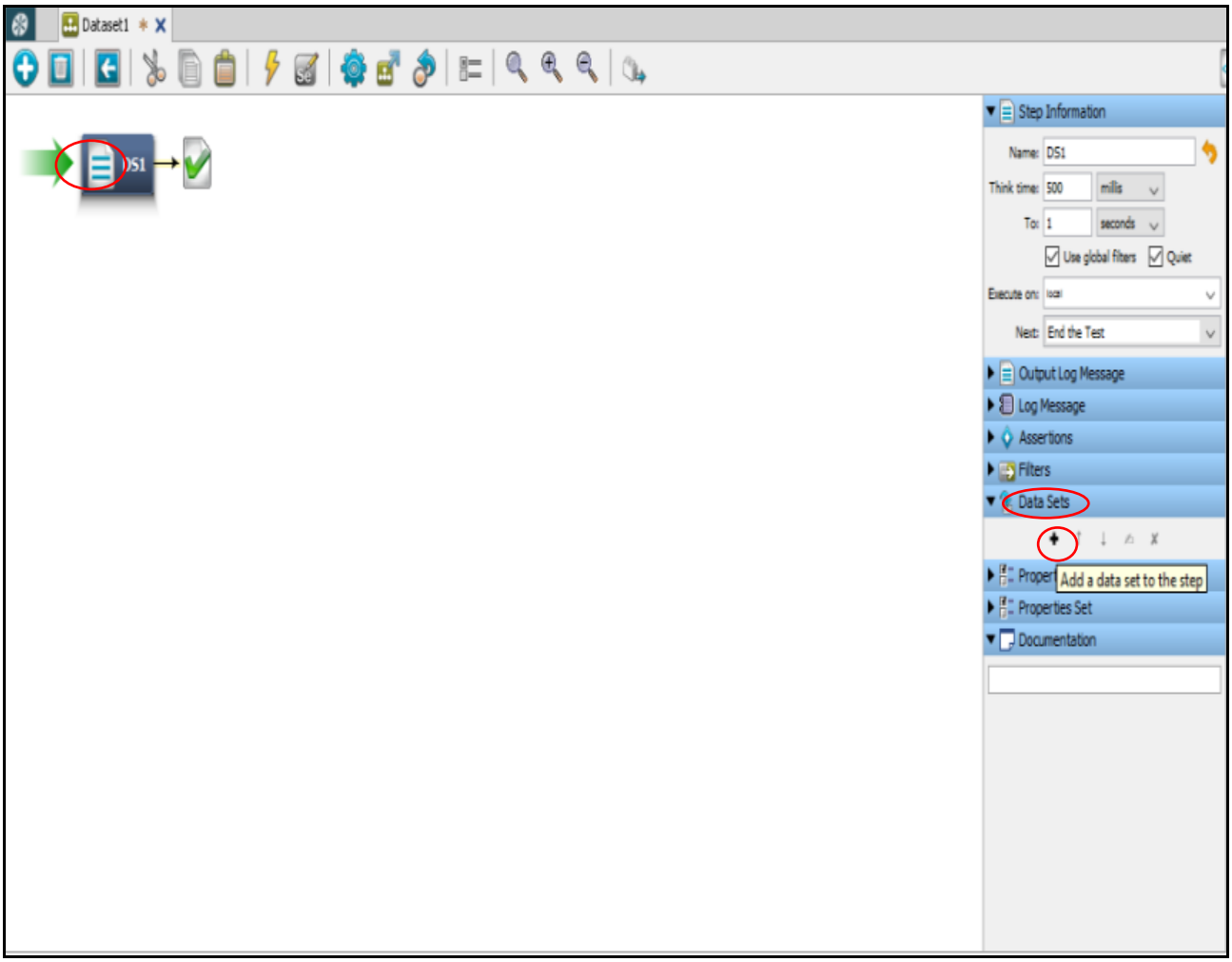


Notice that the month, day, and year properties have not been replaced with actual values. This result is expected, because we have not added the data set to the test case

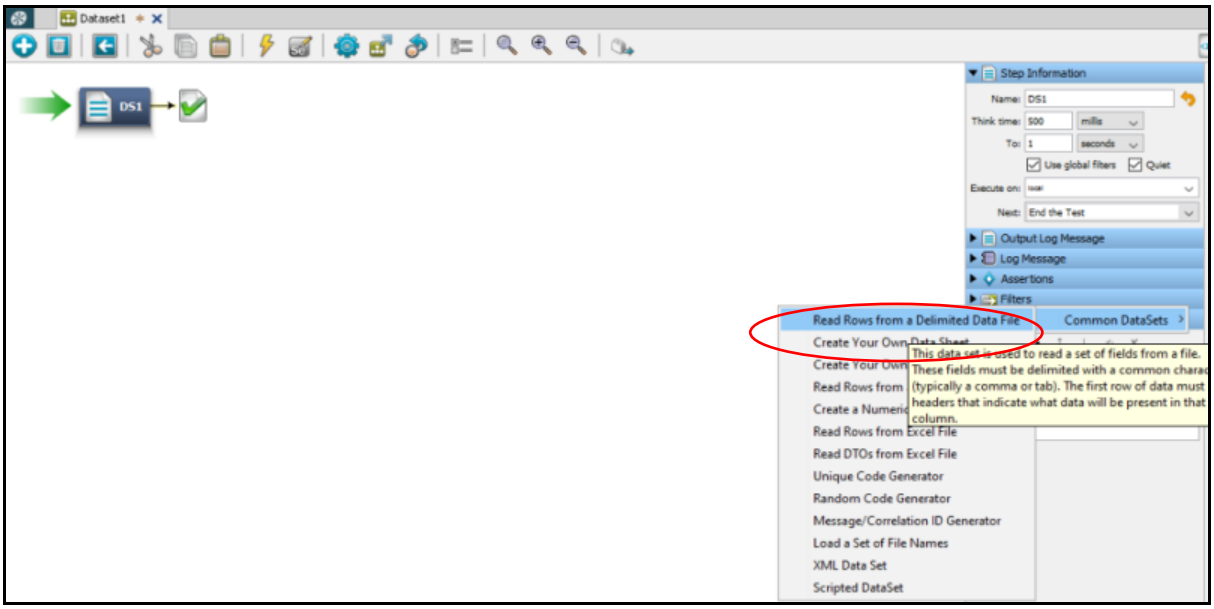
- Create comma-delimited text file as the data set. This option is only one of several options available to create a data set. Do not use spaces in the text file



- Select DSstep1 > double-click the Data Sets > Click Add Data Sets element

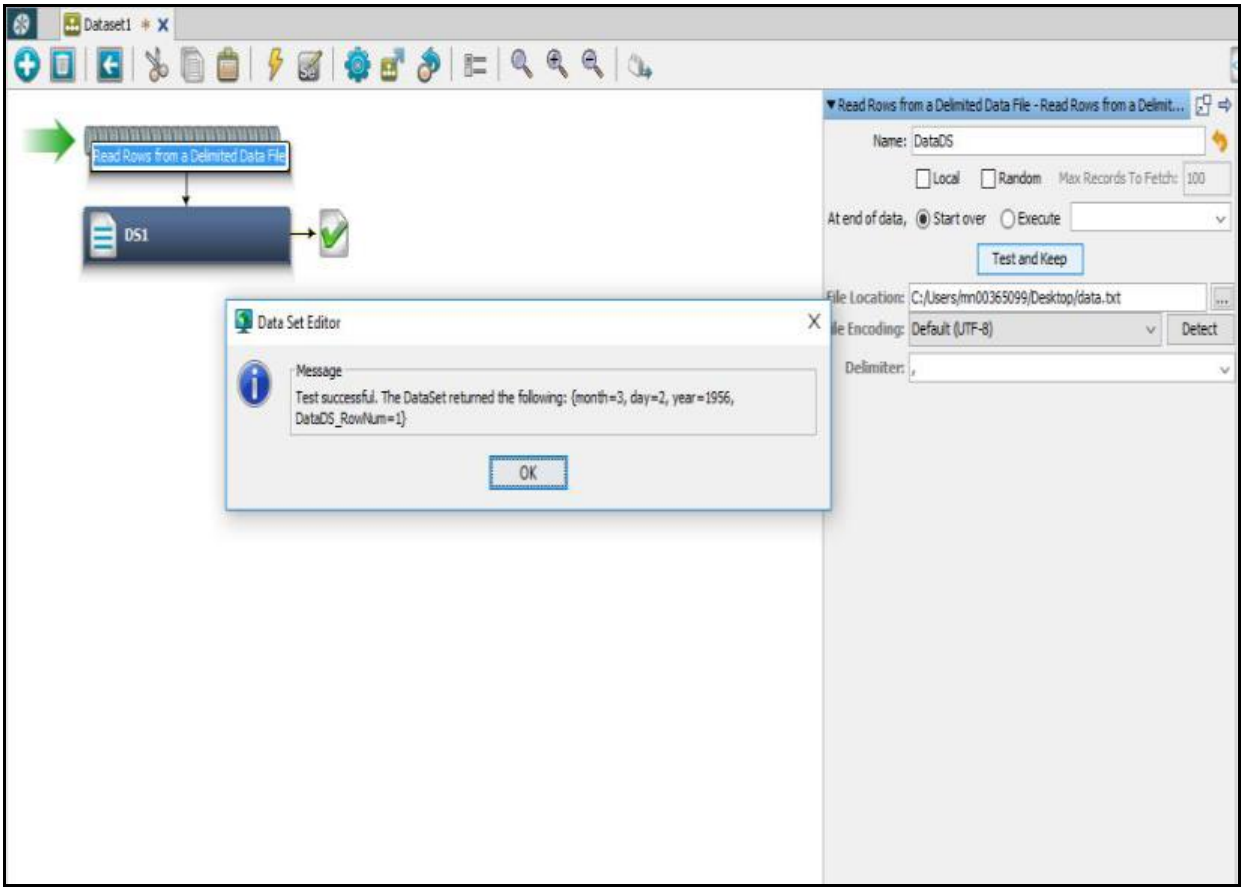


- From Common Datasets list, select **Read Rows from a Delimited Data File**

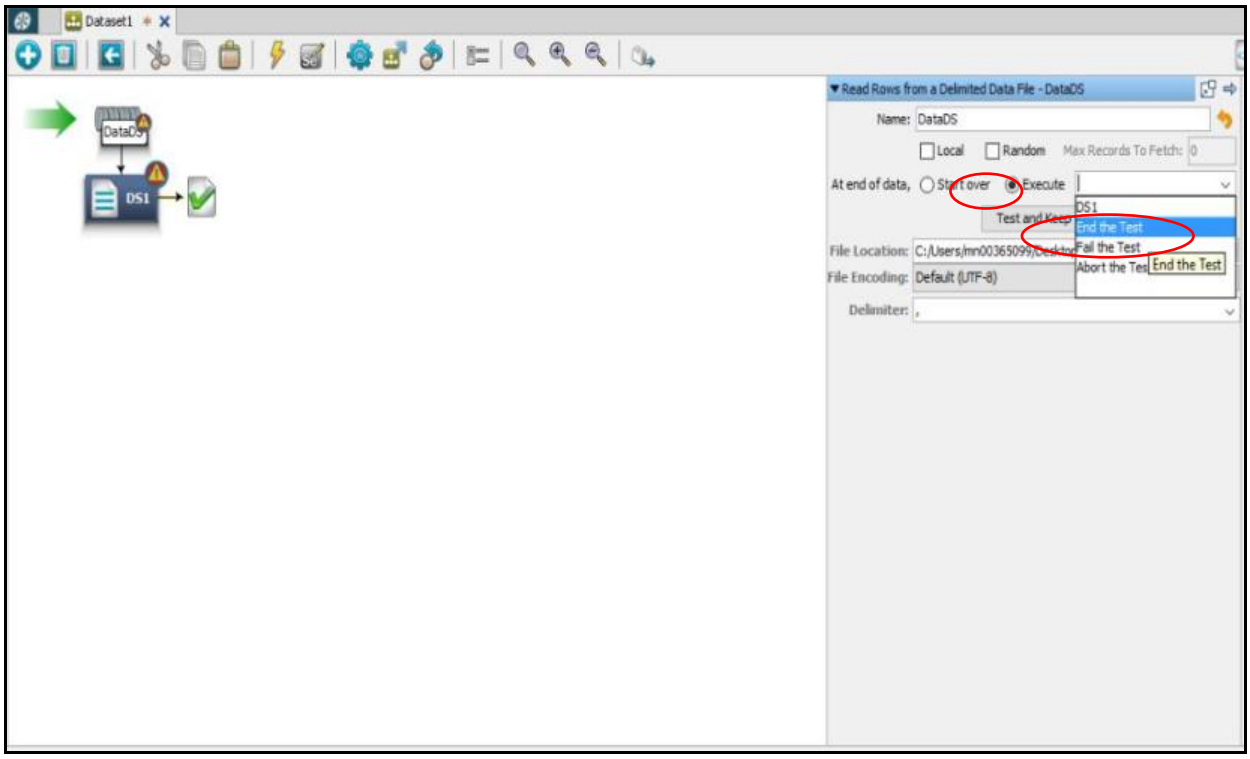


- Set the name to DatesDS. Click File Location and select the dates.txt file.

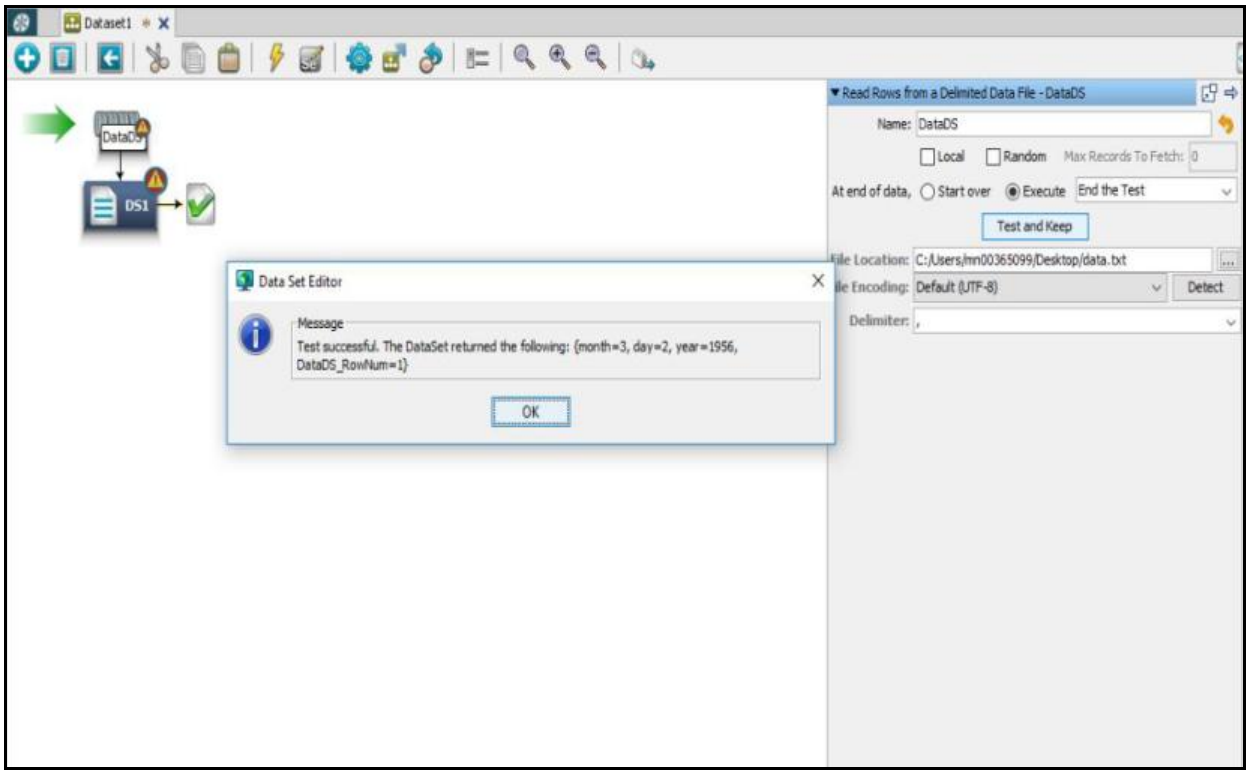
Click the Test and Keep button. If the test is successful, the Data Set Editor window returns a "Test successful" message. Click OK button.



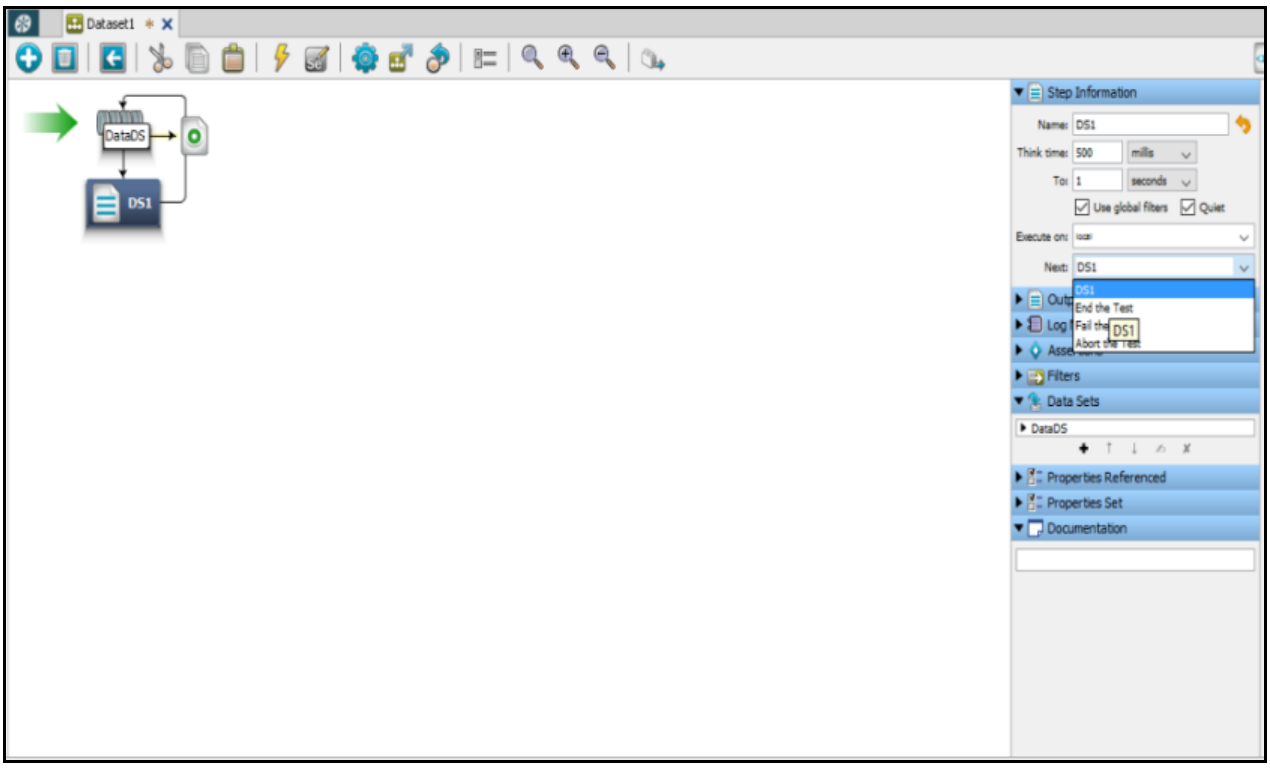
- At end of data field, select the Execute option. Click the drop-down arrow and select End the Test



- Click the **Test and Keep** button and OK button

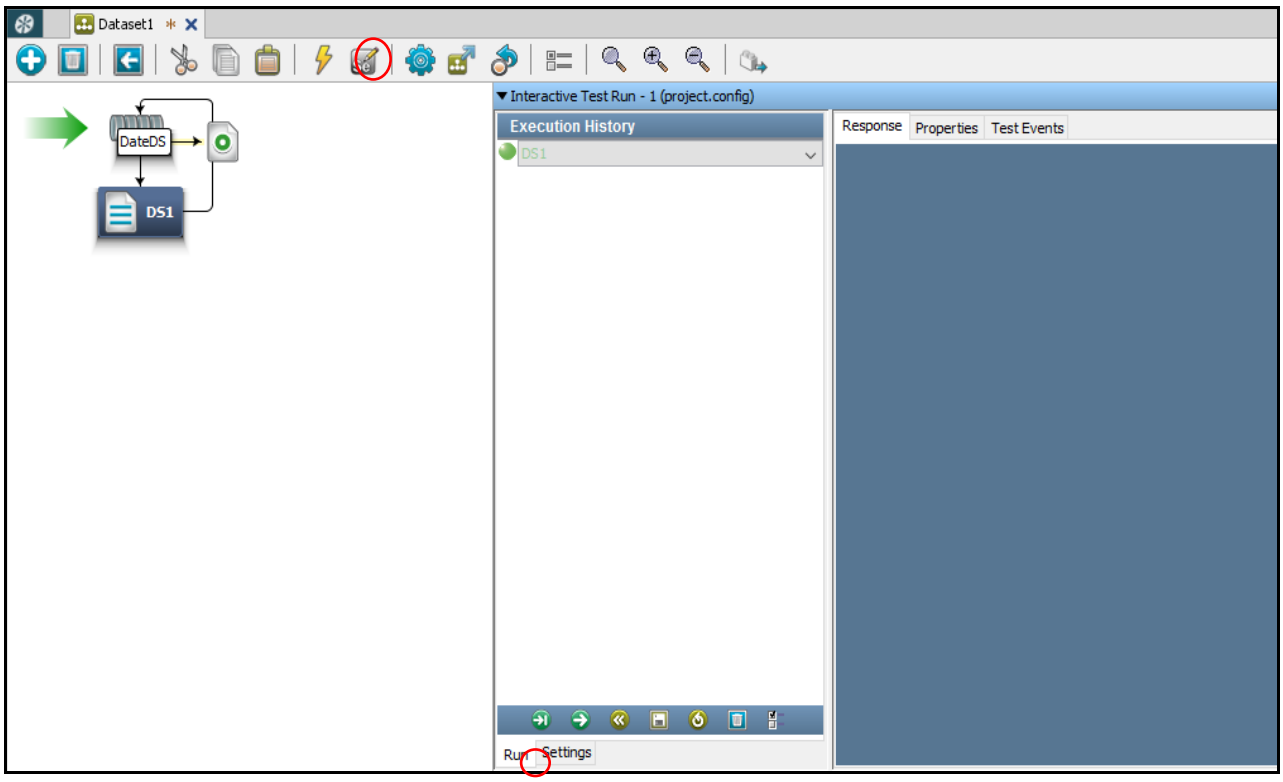


- Select the DS1 > Set the **Next** drop-down list to DS1.

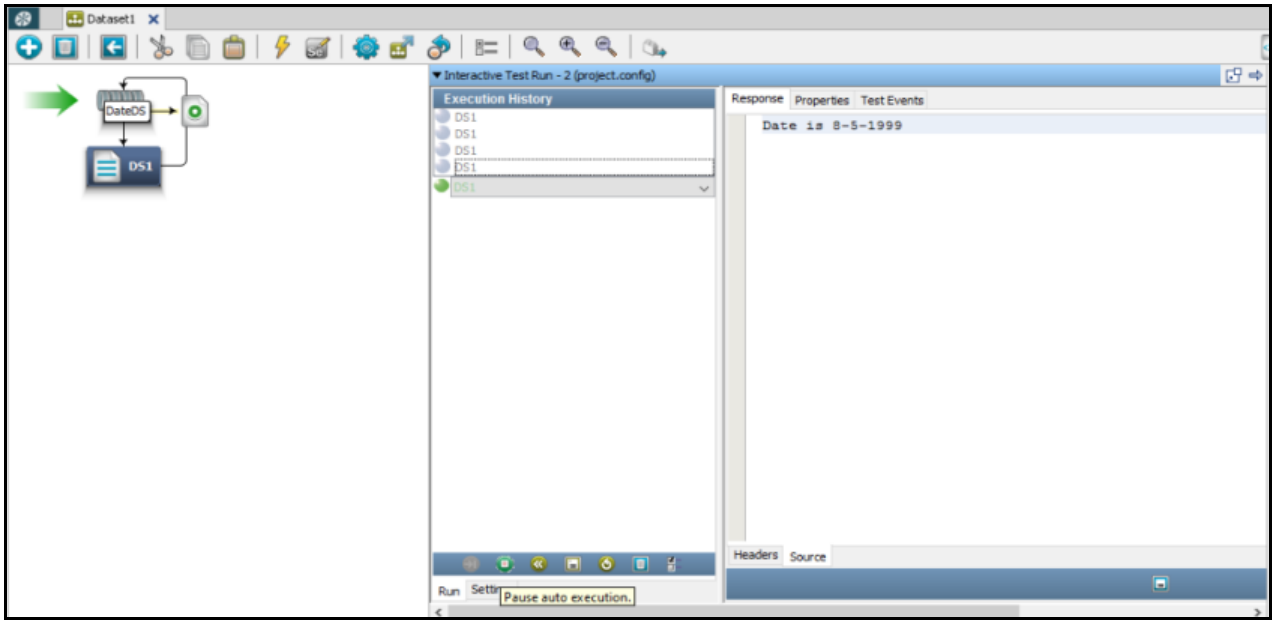


This setting causes the step to loop. The arrows in the model editor show the order of execution: DS1 followed by DatesDS.

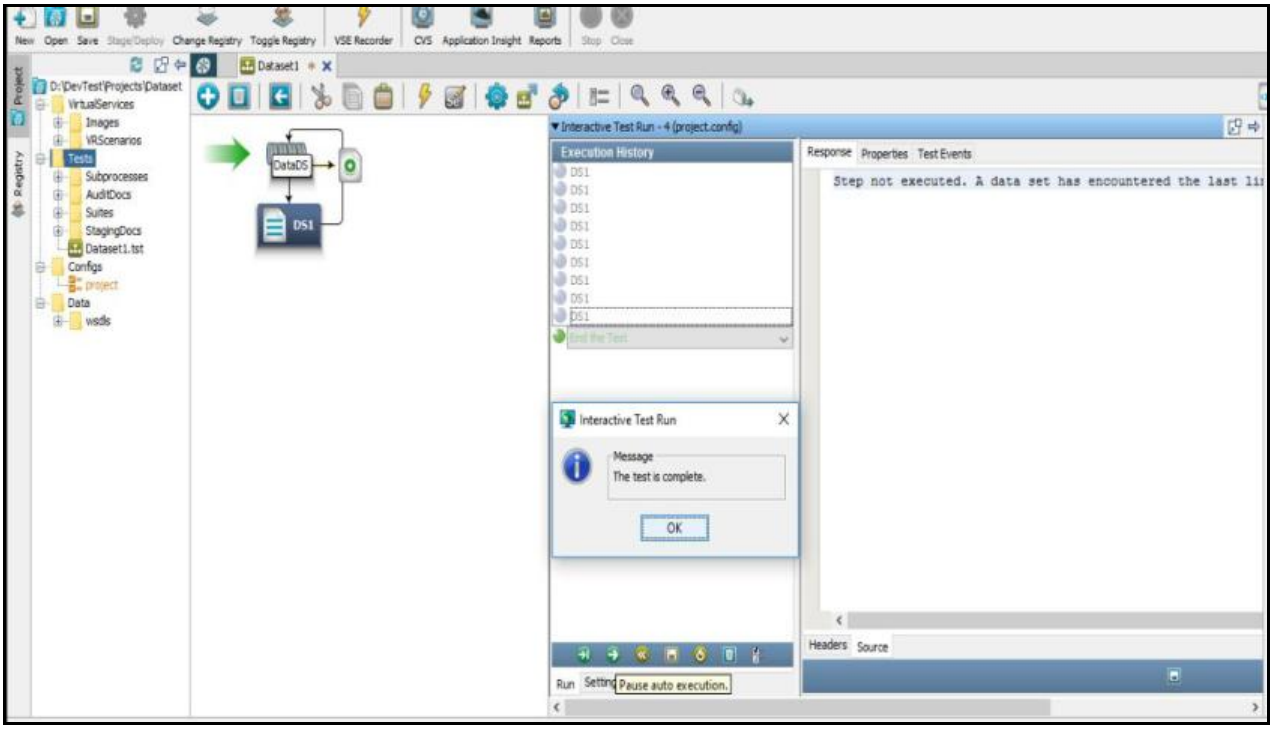
- Start ITR and click Automatically execute test



- The test case runs in a loop until there are no more data rows in the data set.



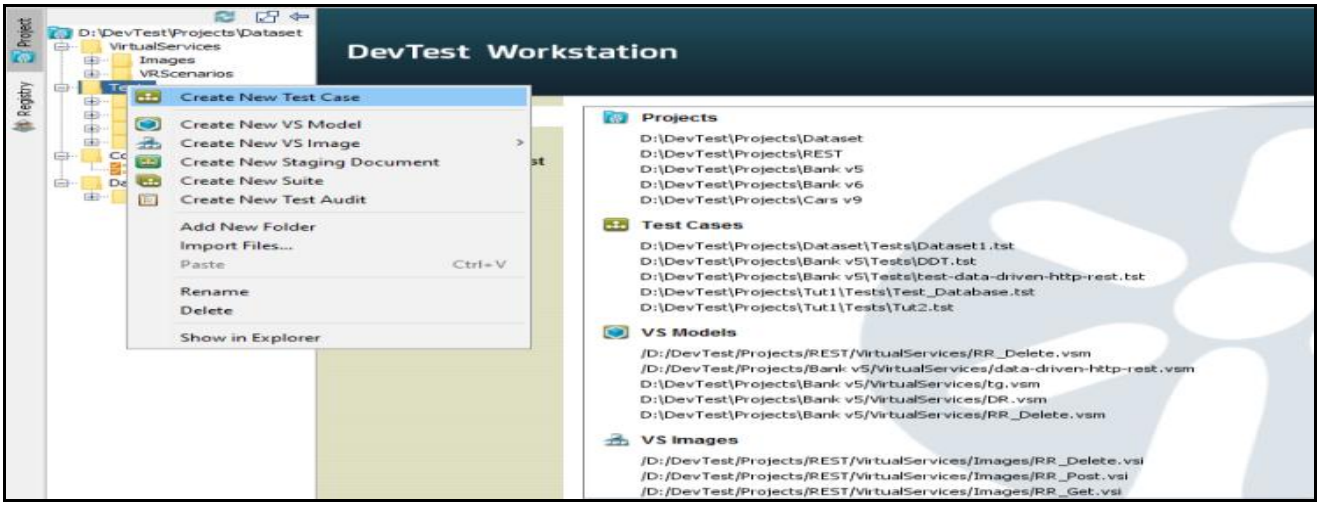
- Test is complete, click OK button and Save



21. Connecting DevTest to Internal Database.

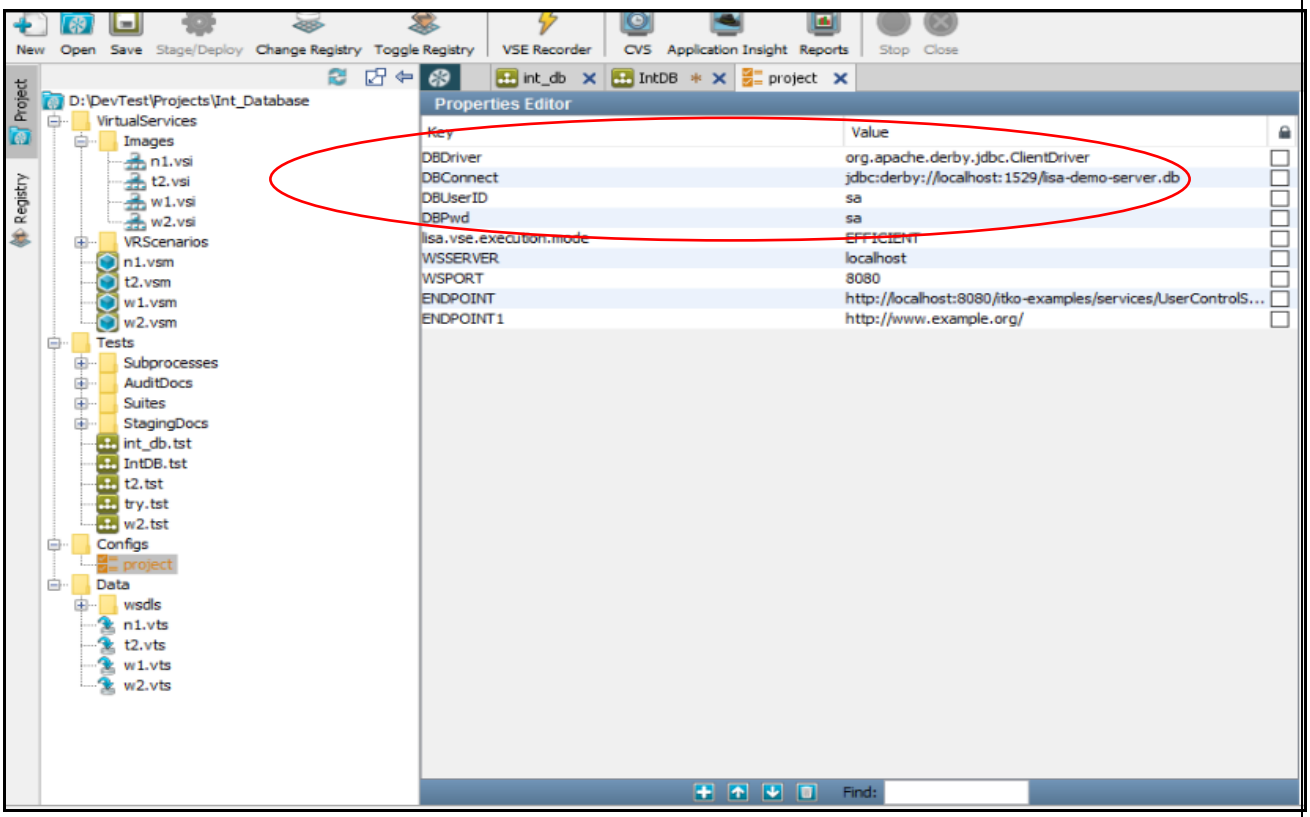
Following are the Steps for Connecting to Internal Database, creating, inserting and executing a SQL query against the database, Adding assertions and filters.

1) Projects > Test case > Create New Test Case

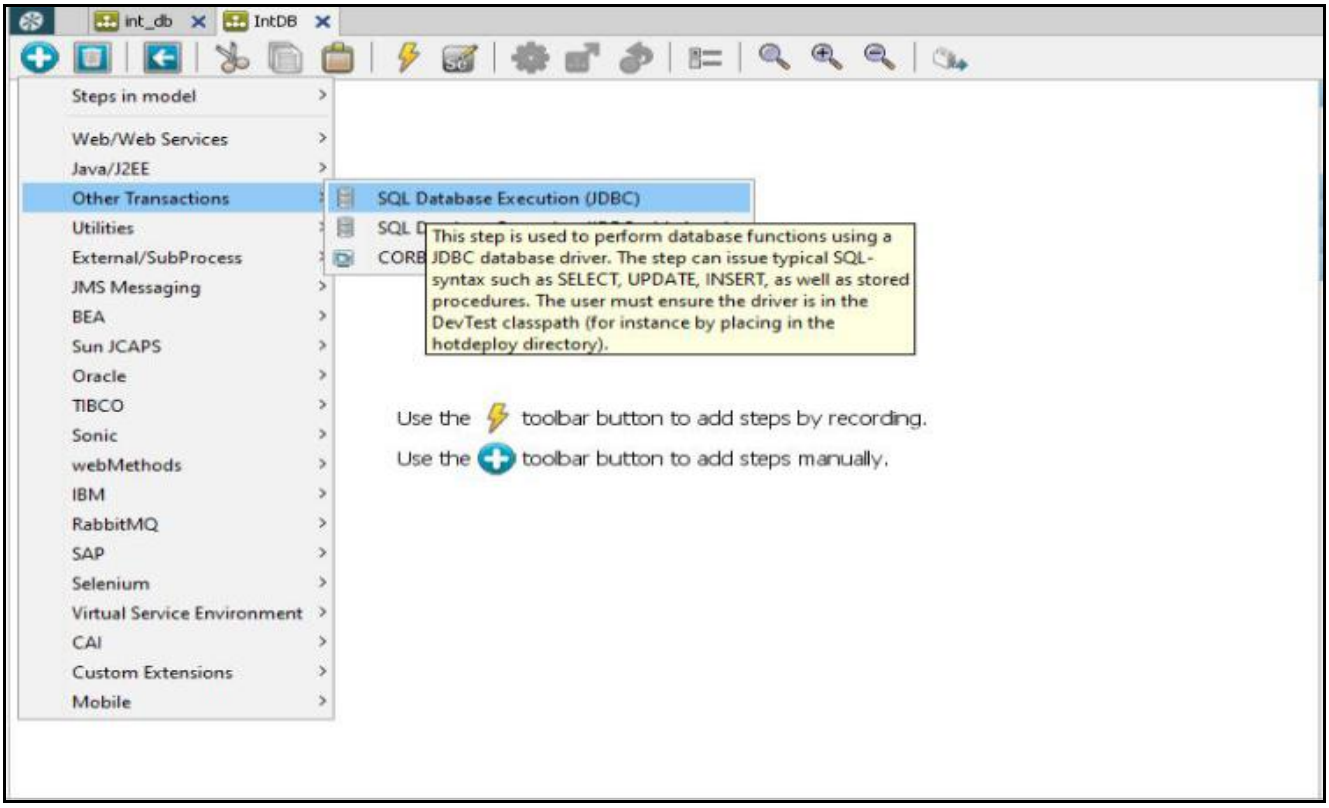


2) Configs > projects > Add following Database properties to the configuration.

- DBDriver: org.apache.derby.jdbc.ClientDriver
- DBConnect: jdbc:derby://localhost:1529/lisa-demo-server.db
- DBUserID: XX
- DBPwd: XX



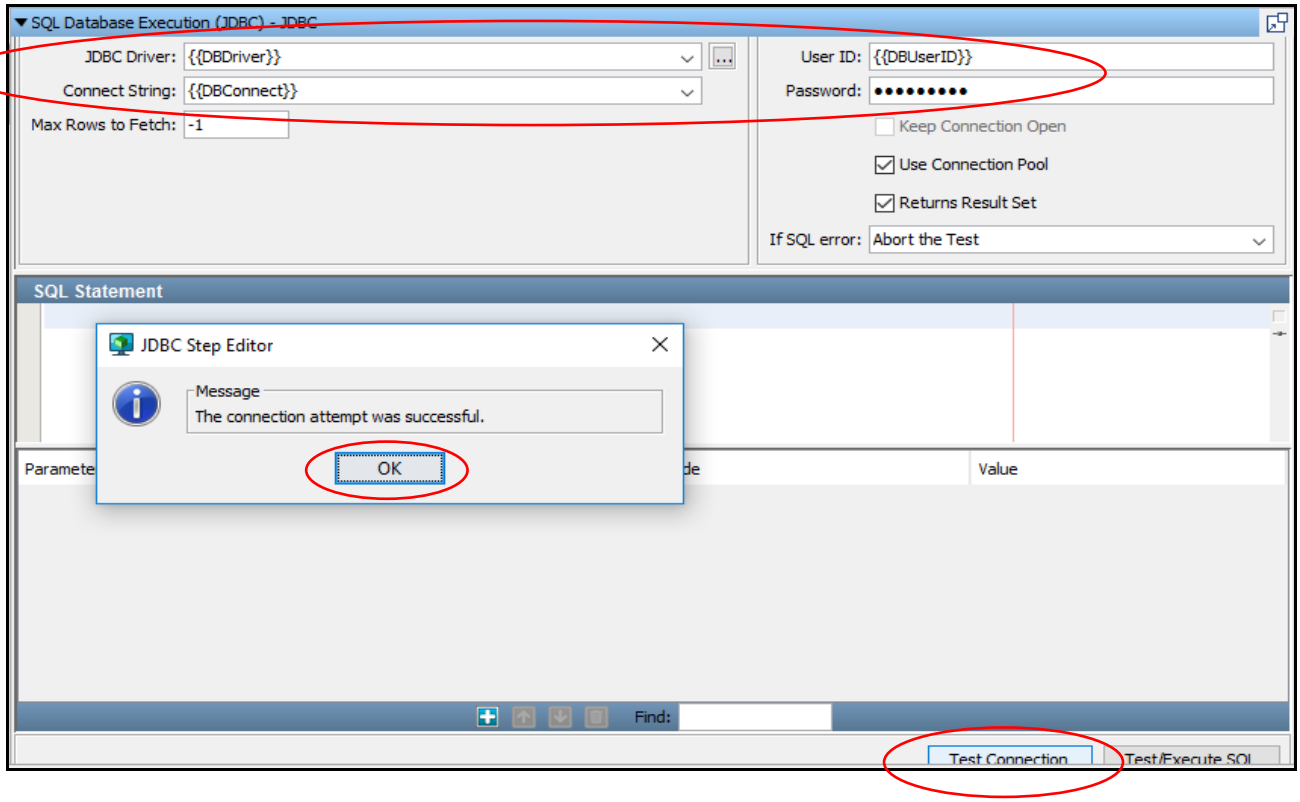
3) Click  Add > Select Other transactions and SQL Database Execution(JDBC).



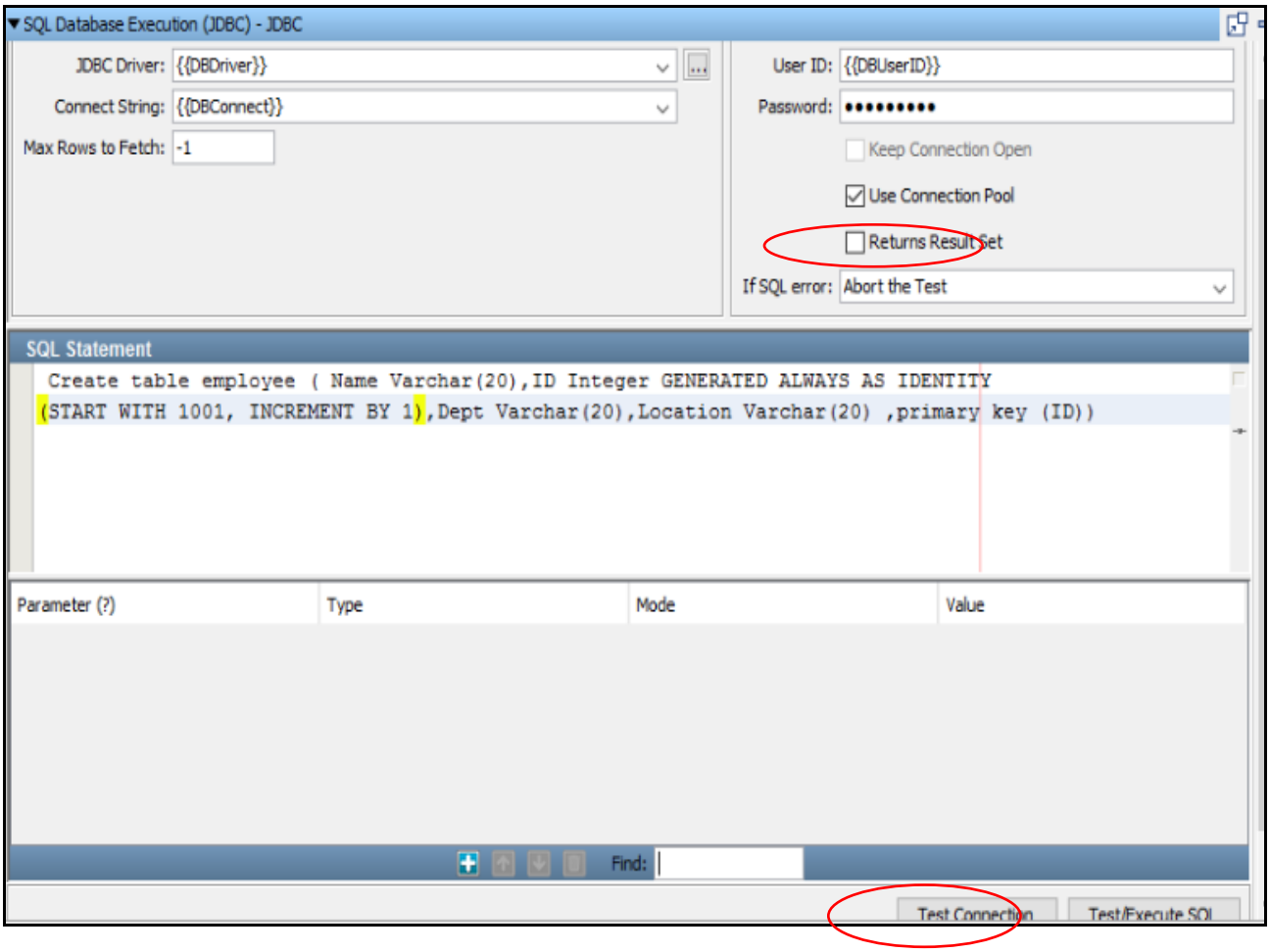
4) Double click on JDBC > enter the following to connect to database

- JDBC Driver: {{DBDriver}}
- Connect String: {{DBConnect}}
- User ID: {{DBUserID}}
- Password: {{DBPwd}}

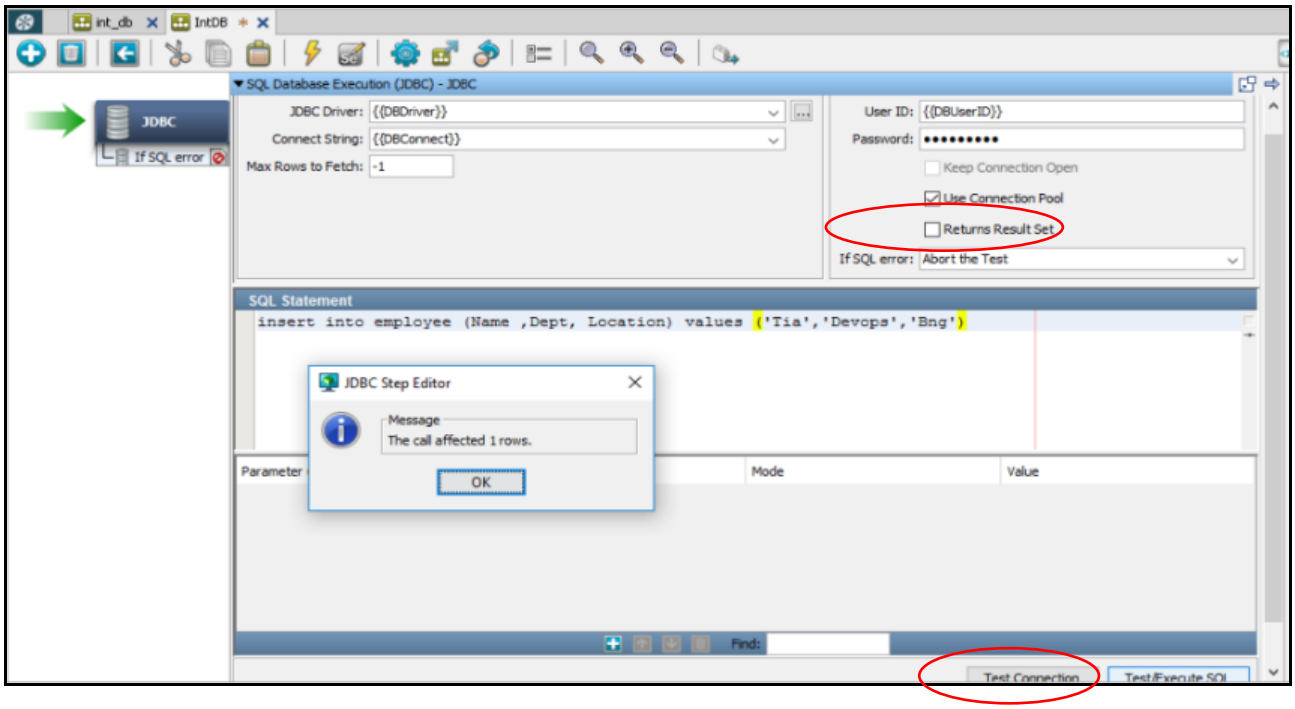
Click on Test connection


SQL Database Execution (JDBC) - JDBC
JDBC Driver: {{DBDriver}}
Connect String: {{DBConnect}}
Max Rows to Fetch: -1
User ID: {{DBUserID}}
Password: ••••••••••
Keep Connection Open
Use Connection Pool
Returns Result Set
If SQL error: Abort the Test
SQL Statement
JDBC Step Editor
Message
The connection attempt was successful.
OK
Value
Find:
Test Connection
Test/Execute SQL

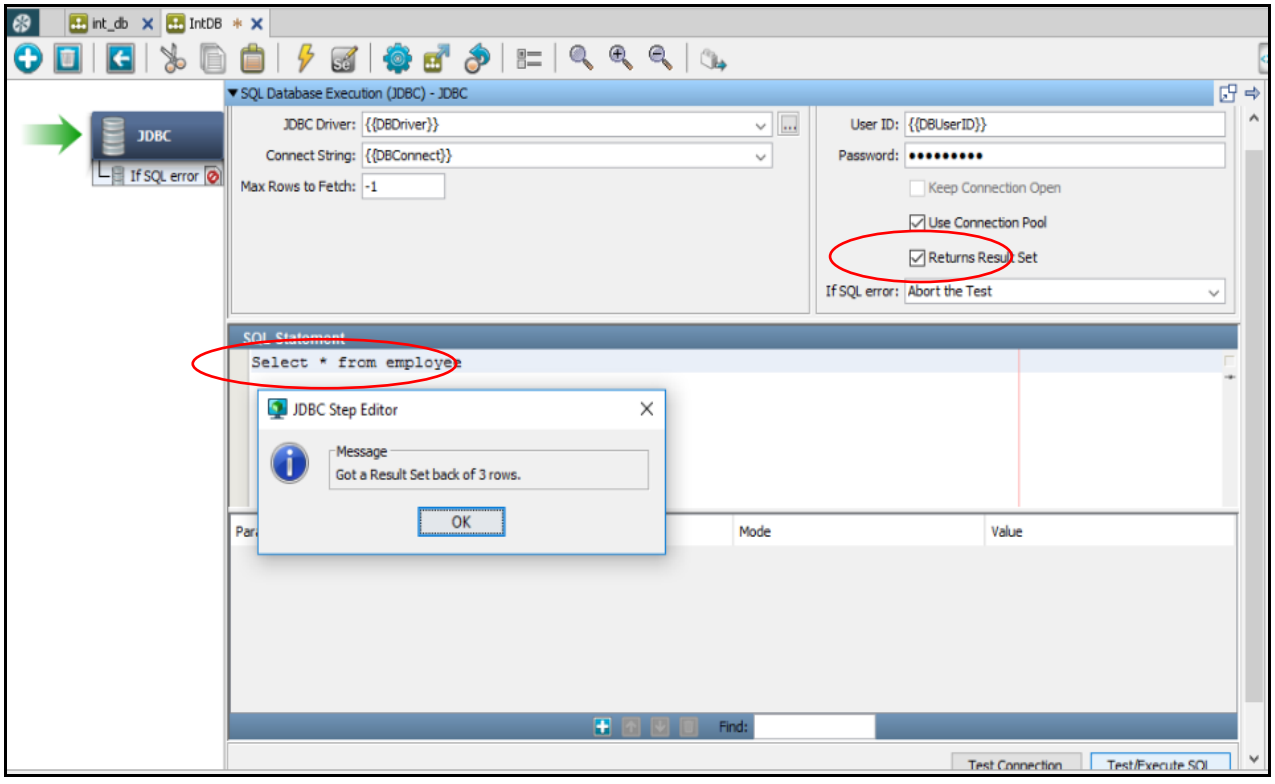
5) Create a Table. While creating uncheck the Return Result set.

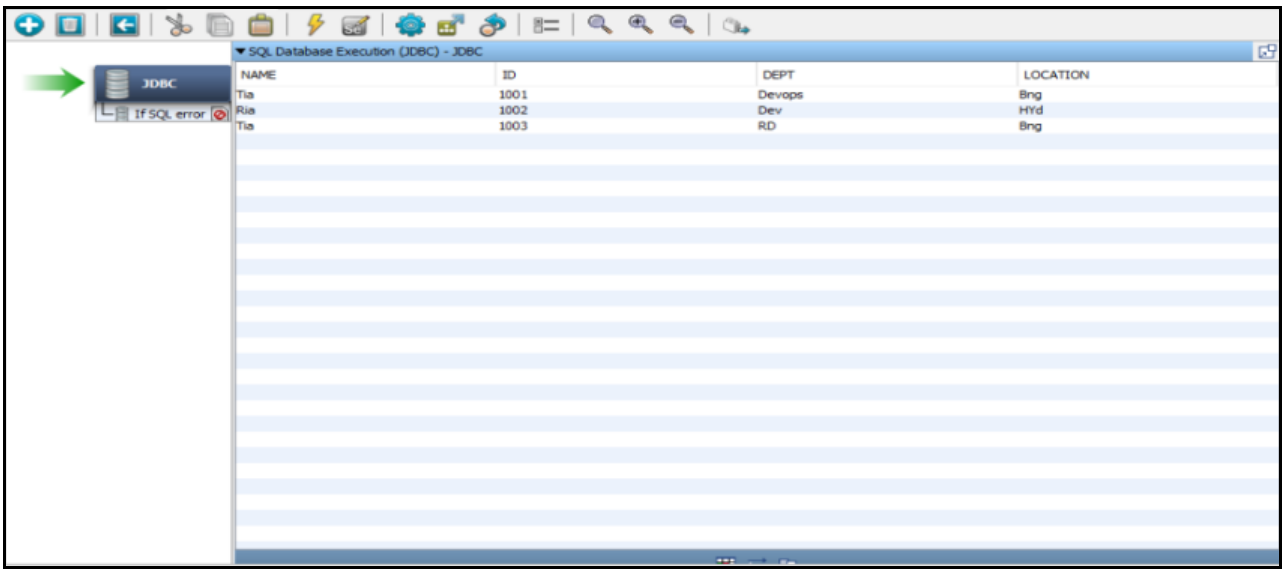


6) Insert the values created into the table. Uncheck Return Result set. Click on Test /Execute SQL



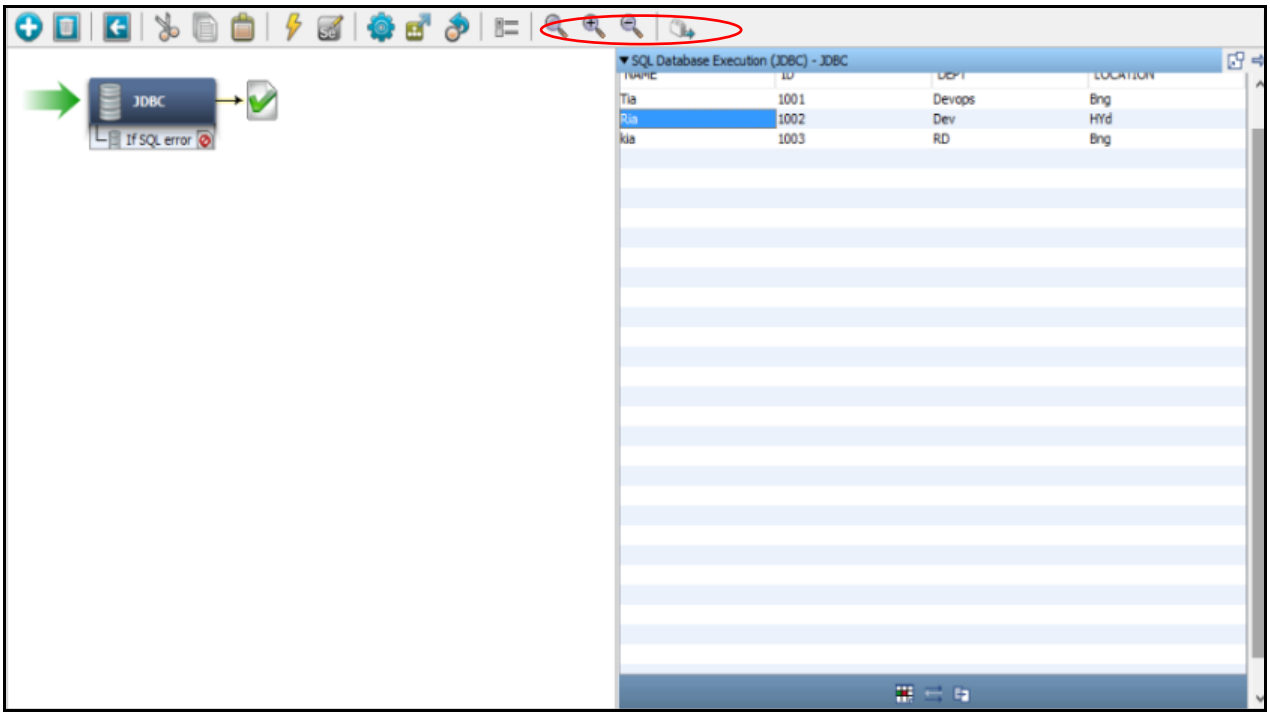
7) Retrieve the values from the table. Check Return Result Set



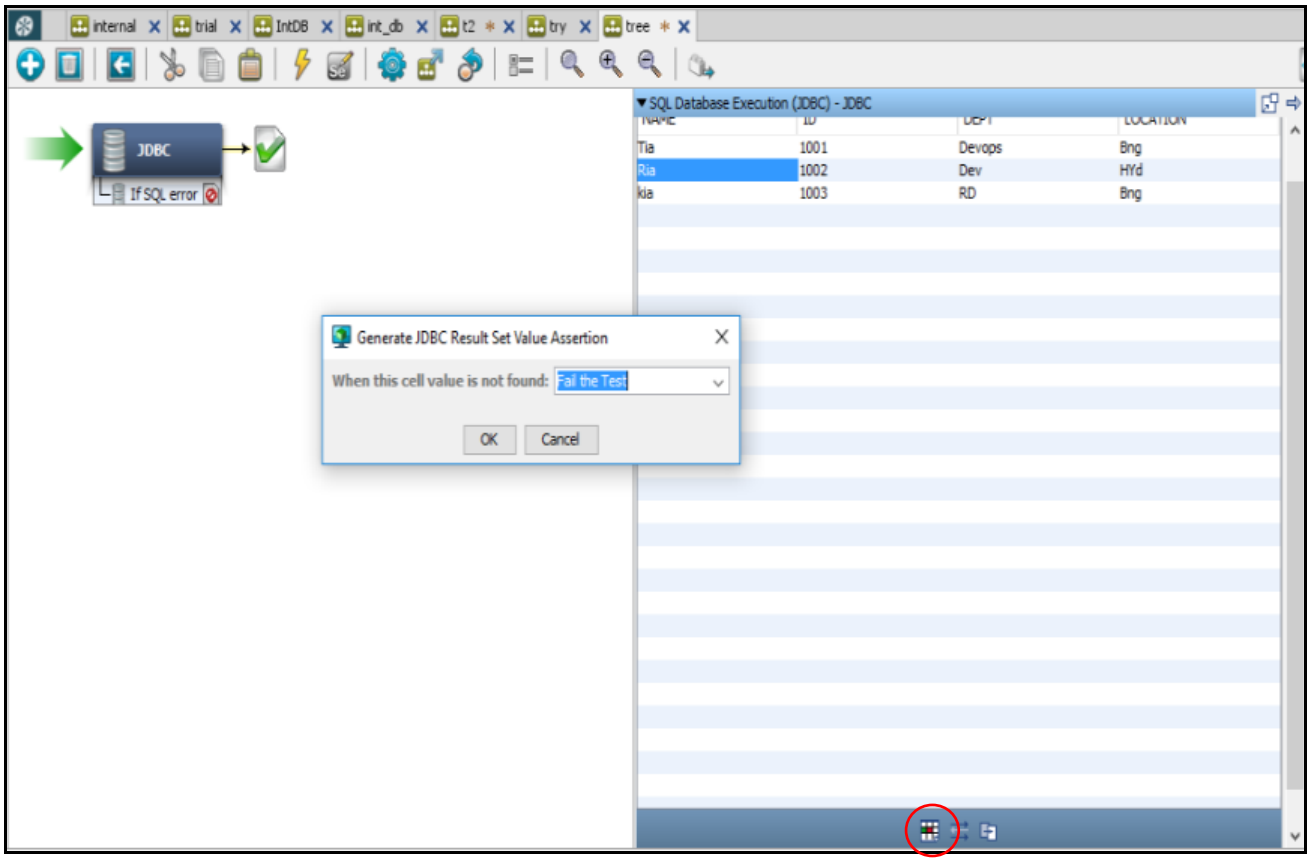


8) Add an assertion that tests for the presence of a specific name in the result set. In the Result Set tab, select a cell in the NAME column.

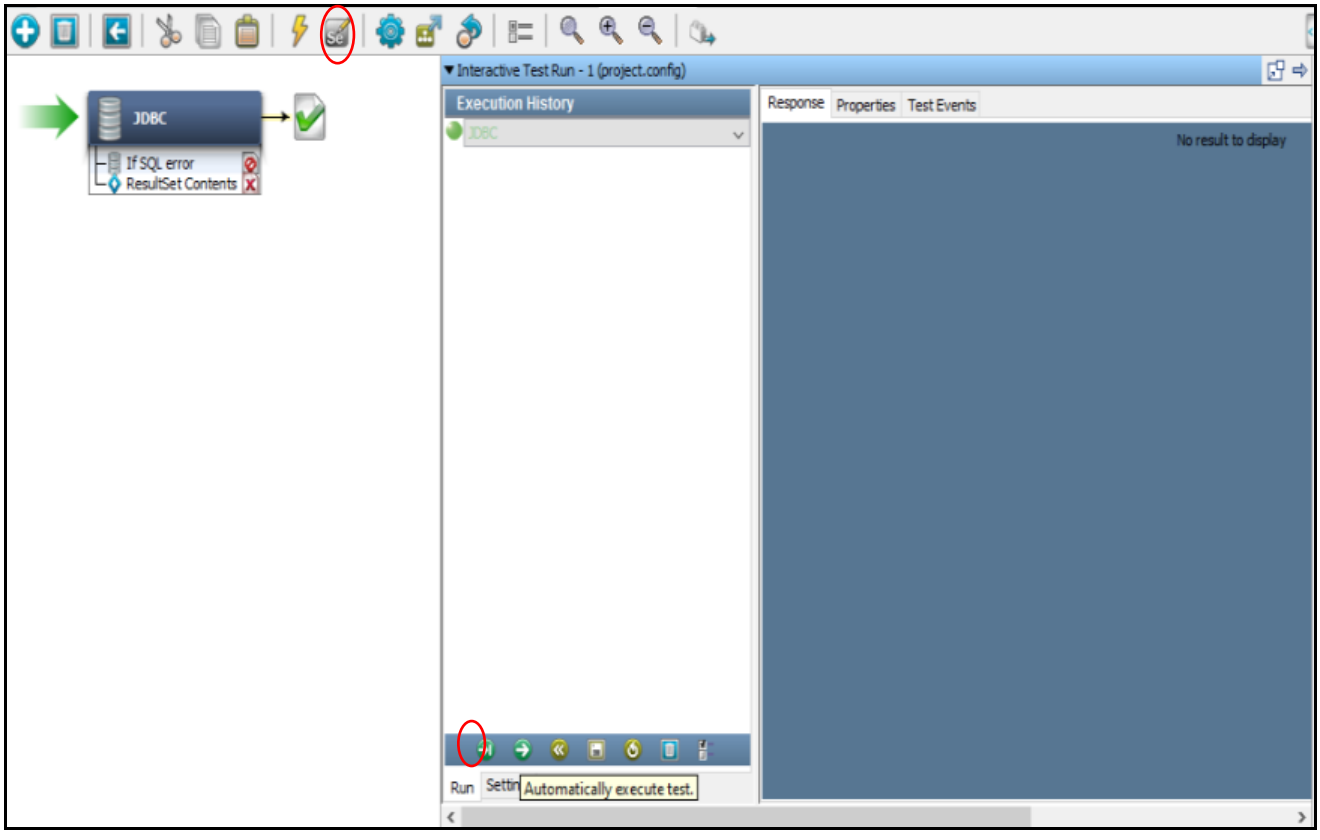
Click  Generate Assertion for the Value of a Cell.



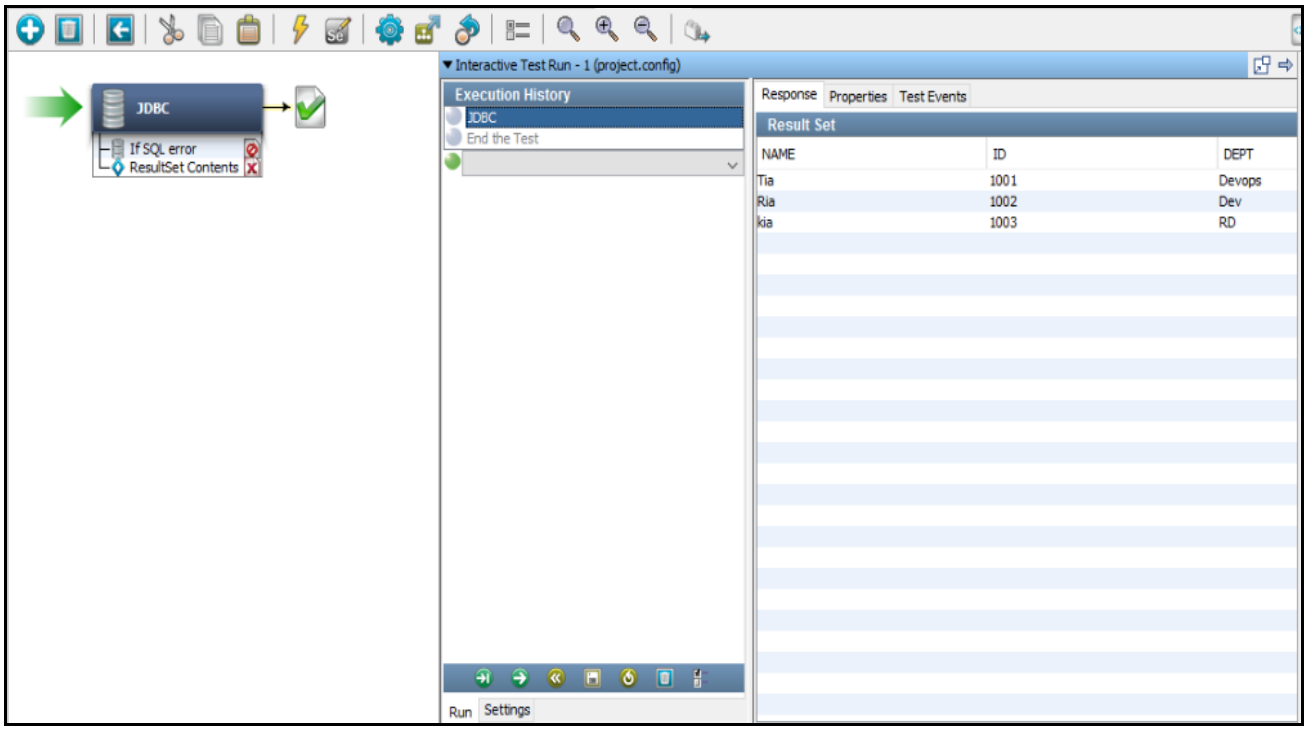
9) In Generate JDBC Result Set Value Assertion > Fail the Test and Click Ok and Save.



10) Start ITR and automatically execute test

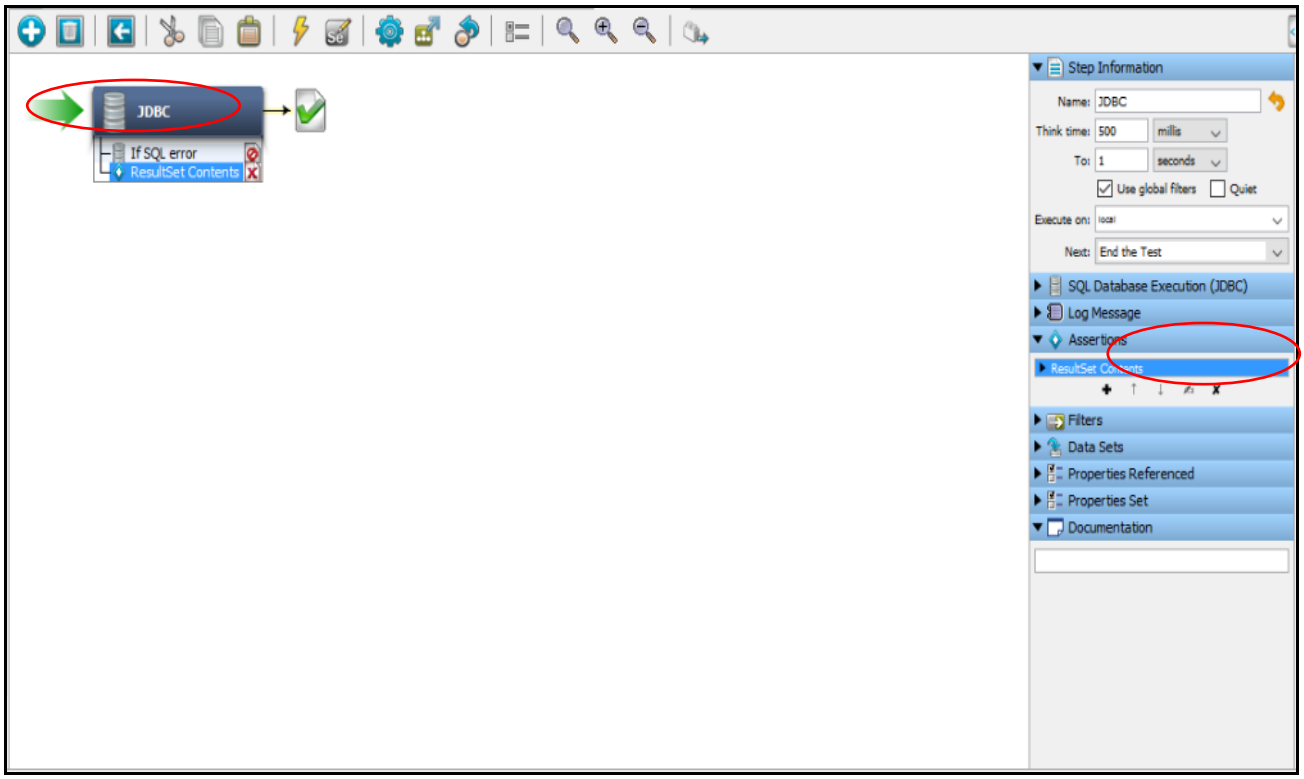


11) The test runs successfully. The result set is shown in the Response tab

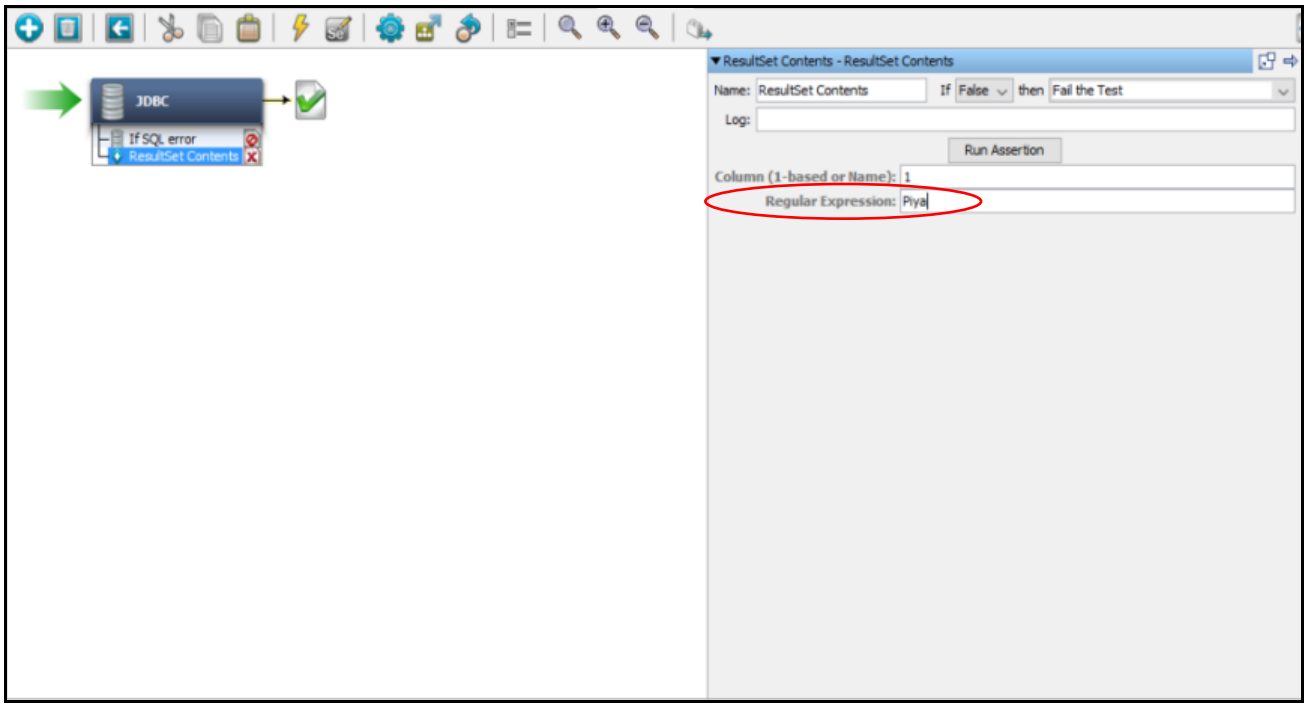


12) Click the JDBC step

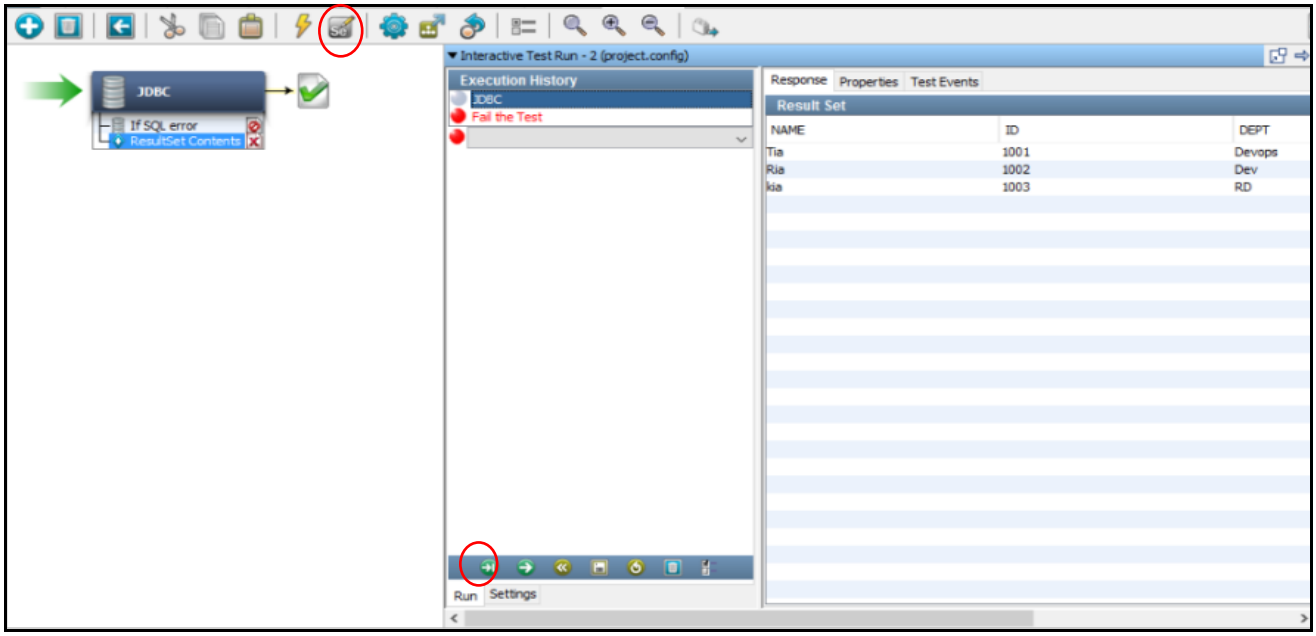
Open the Assertions > Double-click the assertion that created earlier.



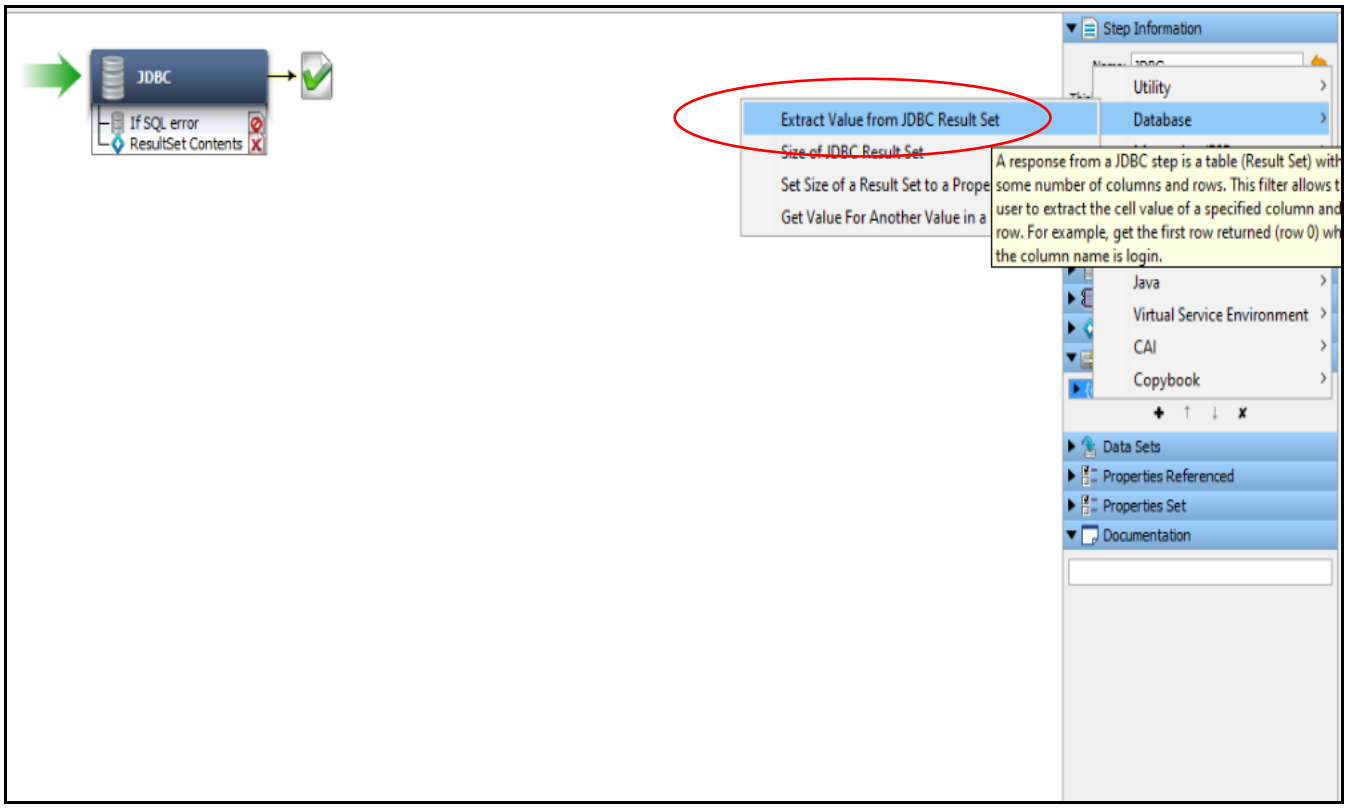
13) Change the value of the **Regular Expression** field and save.



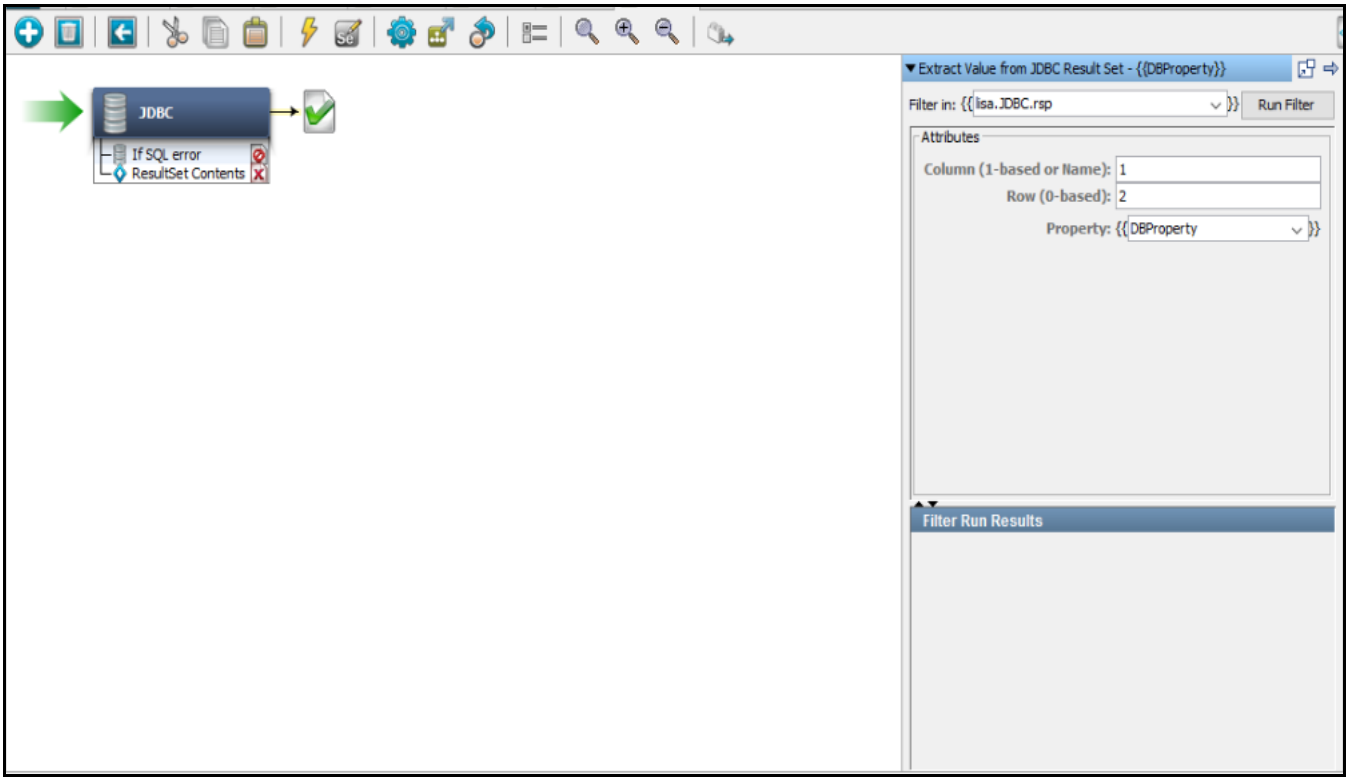
14) Start a new ITR and run the test case again. Test Fails.



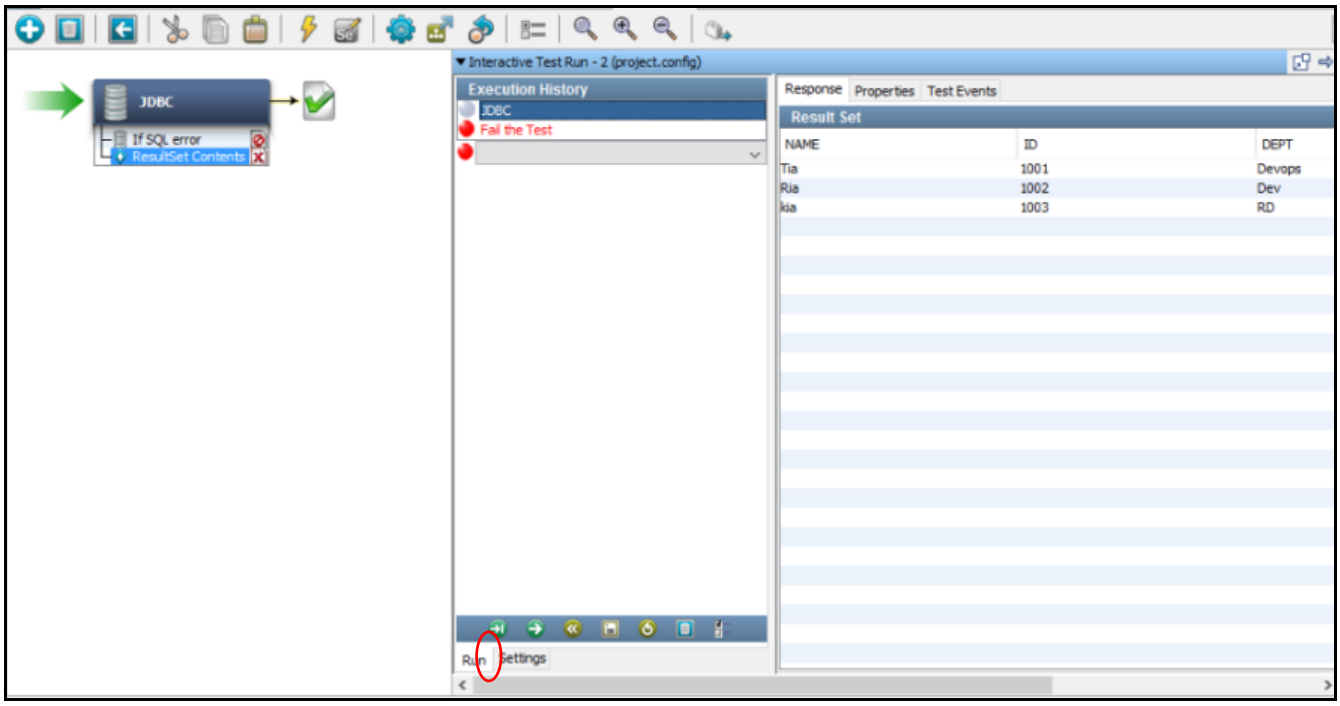
15) Add a database filter that captures the value in the 1st column and 3rd row of the result set. The value is stored in a property. Click the JDBC SELECT Users test step. Open the Filters > Click ✚ Add > Database Filters submenu > select Extract Value from JDBC Result Set.



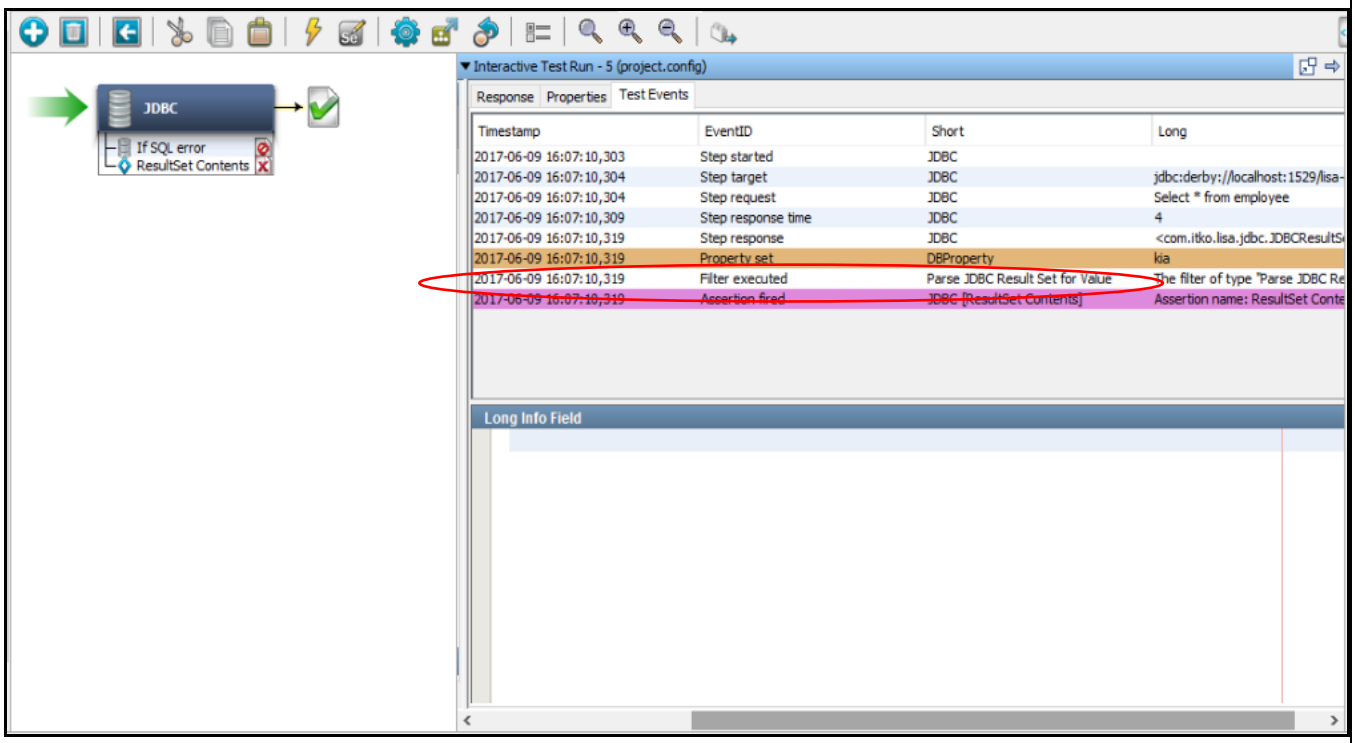
16) Column field, enter 1 Or enter the actual column name, which is Name. In the Row field, enter 2. This field is zero-based. Therefore, the value 2 refers to the 3rd row. In the Property field, enter DBProperty. Click Save.



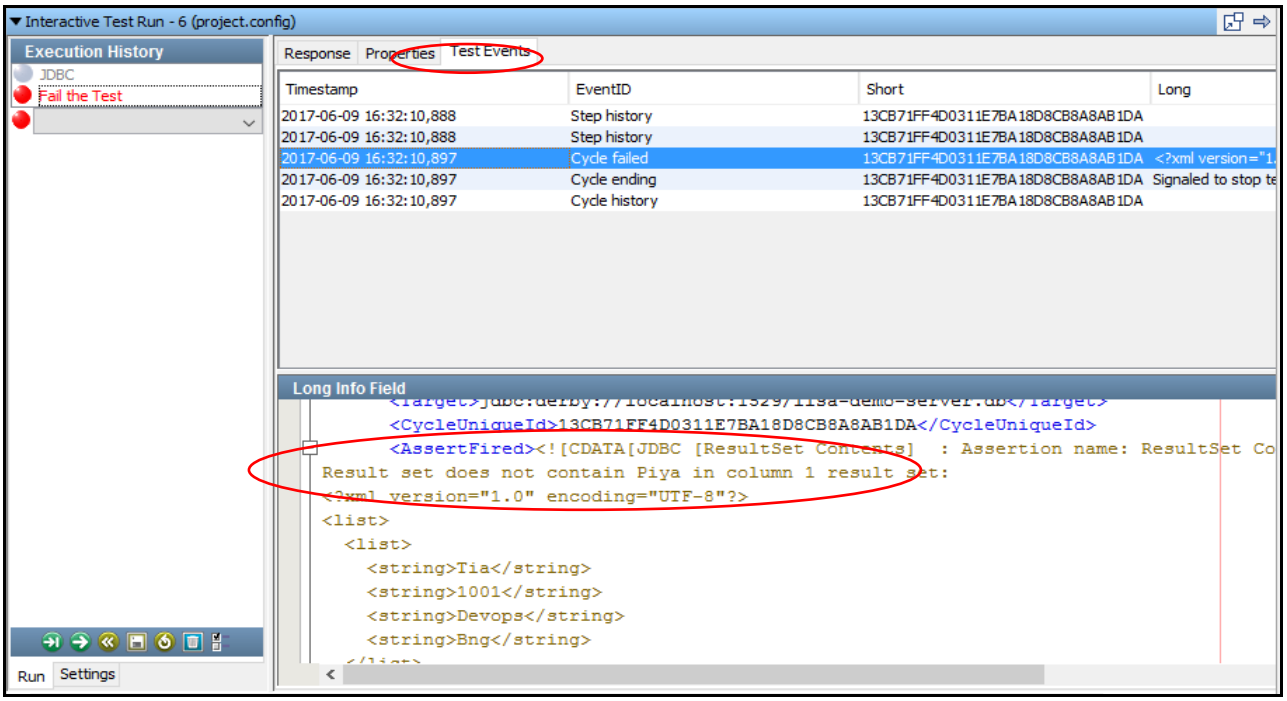
17) Start a new ITR and run the test case again. The test fails because Piya was not found in the result set.



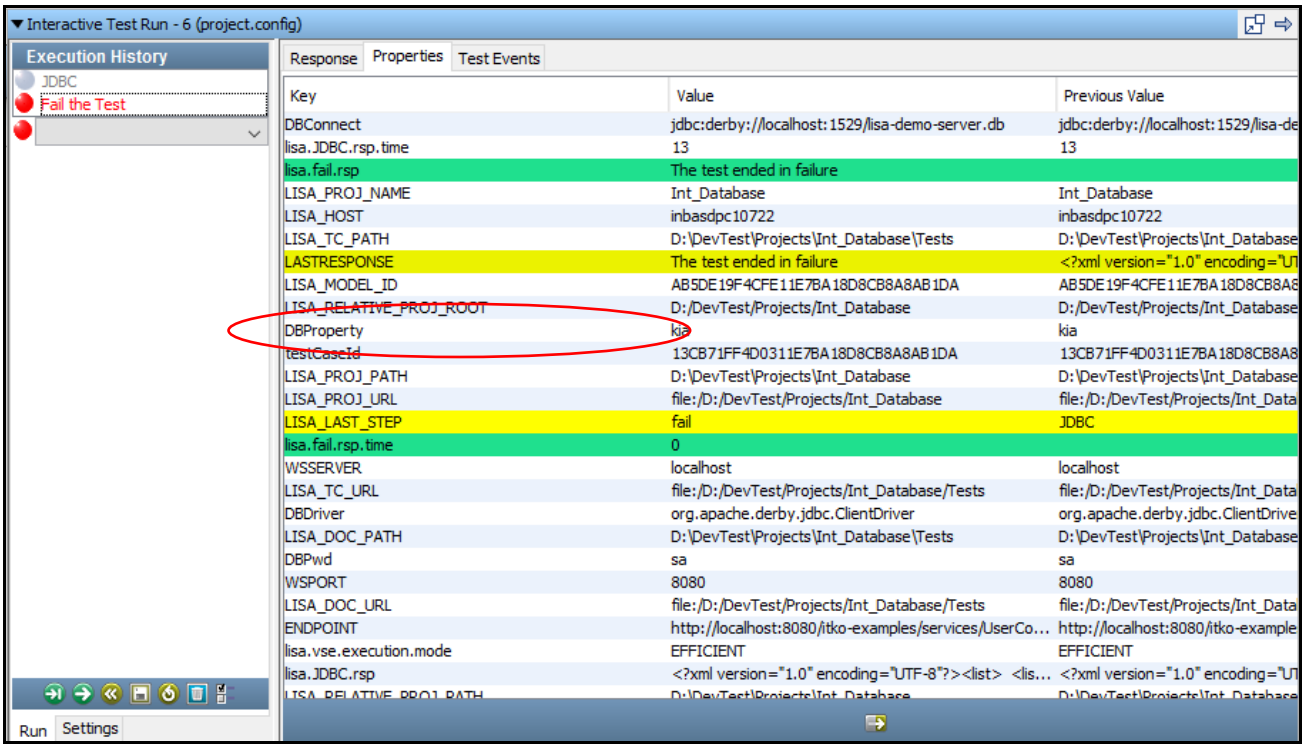
18) Click the Property set event. Notice that DBProperty was set to the value specified by the filter.



19) Click the **Assertion fired** event. The **Long Info Field** area indicates that the assertion fired because the first column of the result set did not contain the value Piya.



20) Click the Properties. Locate and review the DBProperty row.

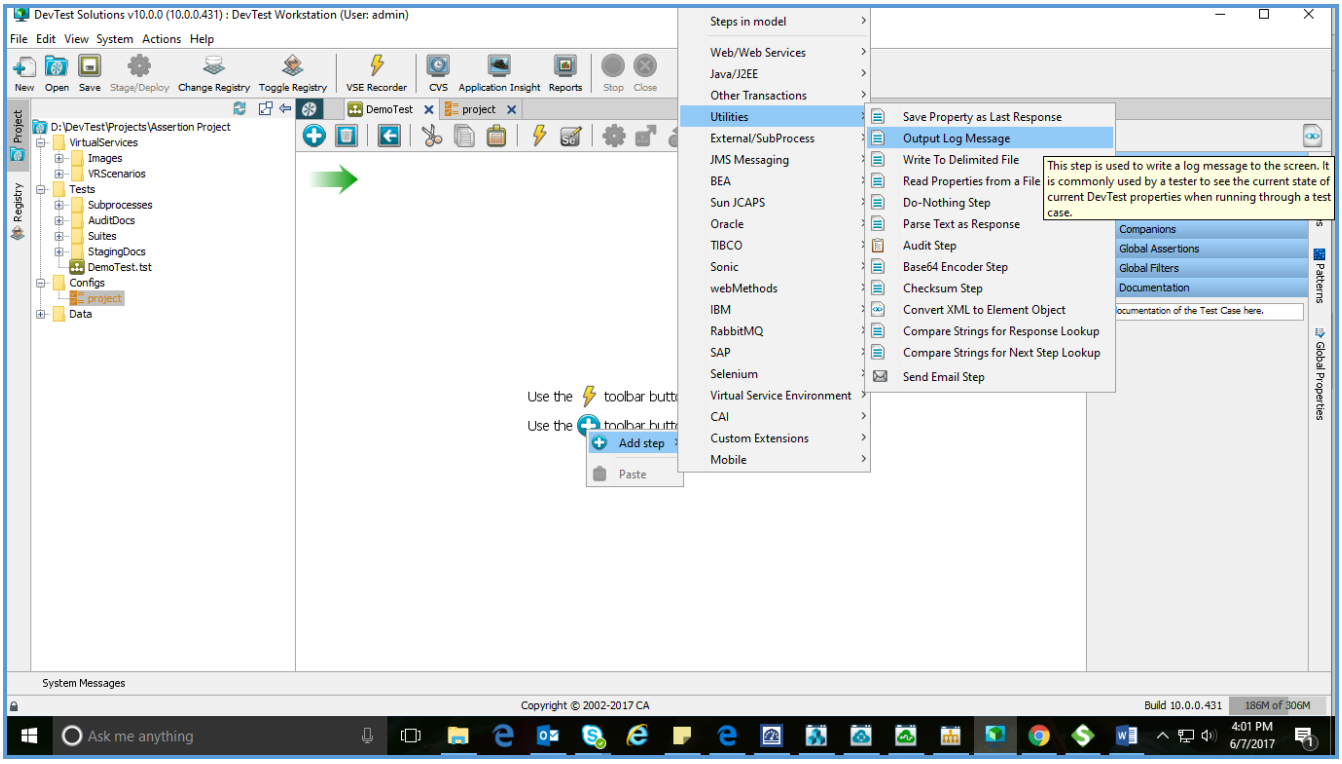


22. Creation and Running Test Case

A test case specifies completely how to test a business component in the "system under test." In some cases, a test case specifies the entire "system under test". A test case is persisted as an XML document, and contains all the information that is necessary to test the component or system. The test cases are created and maintained in DevTest Workstation.

The first step in creating a test case is to create a project. In this project, you can create single or multiple test cases.

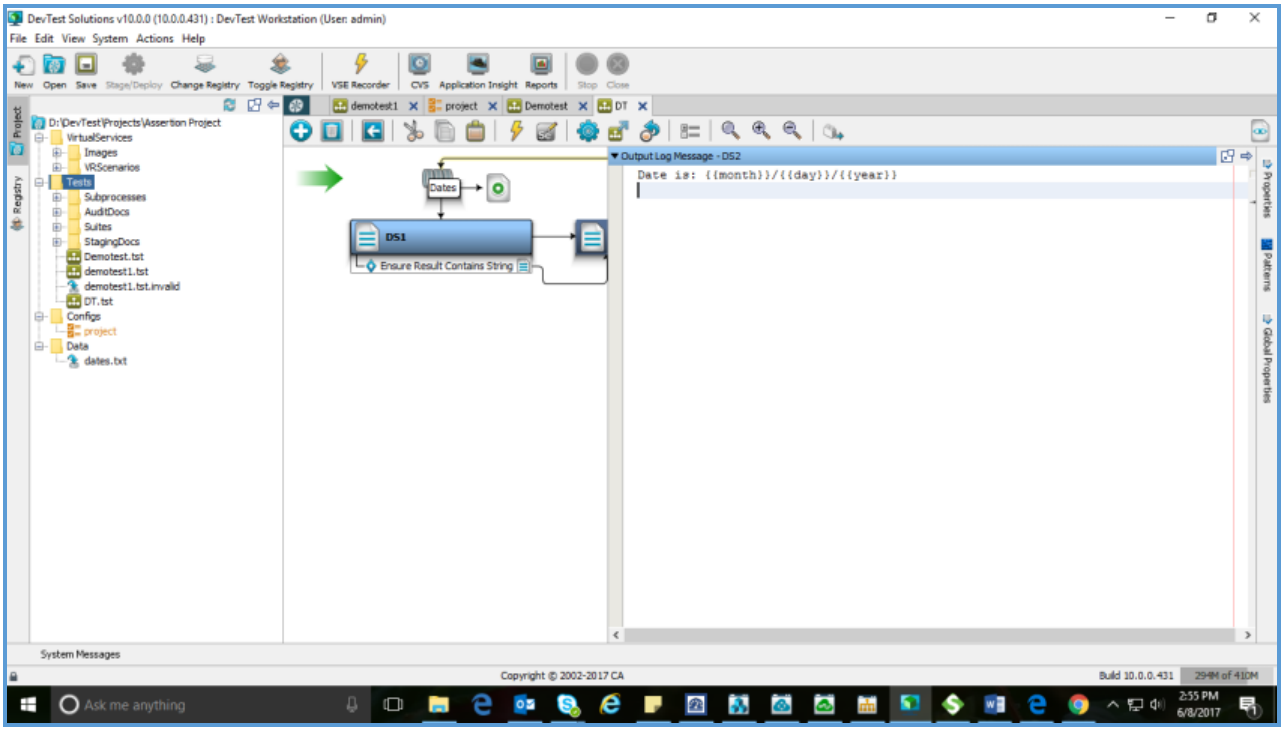
1) Right-click on Test folder to create a new test case in the project panel.
2) Add two steps to the test case. Add **Output Log Message** from **Utilities** and rename by right-click. **DS1** & **DS2** are names used in this example.



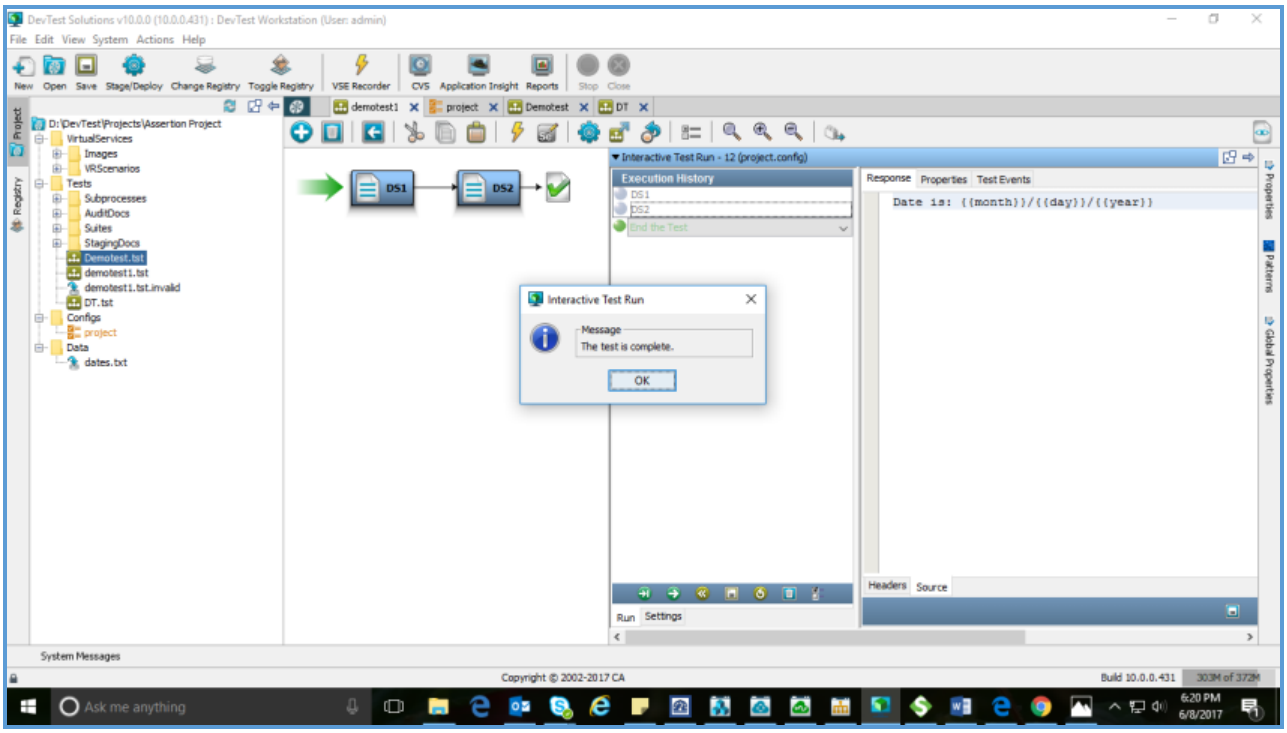
3) Double-click on DS1 in the model editor and change the Output Log Message to

```
Date is: {{month}}/{{day}}/{{year}}
```

4) Repeat step 3 for DS2 and Save



5) Execute the test using the Interactive Test Run (ITR)



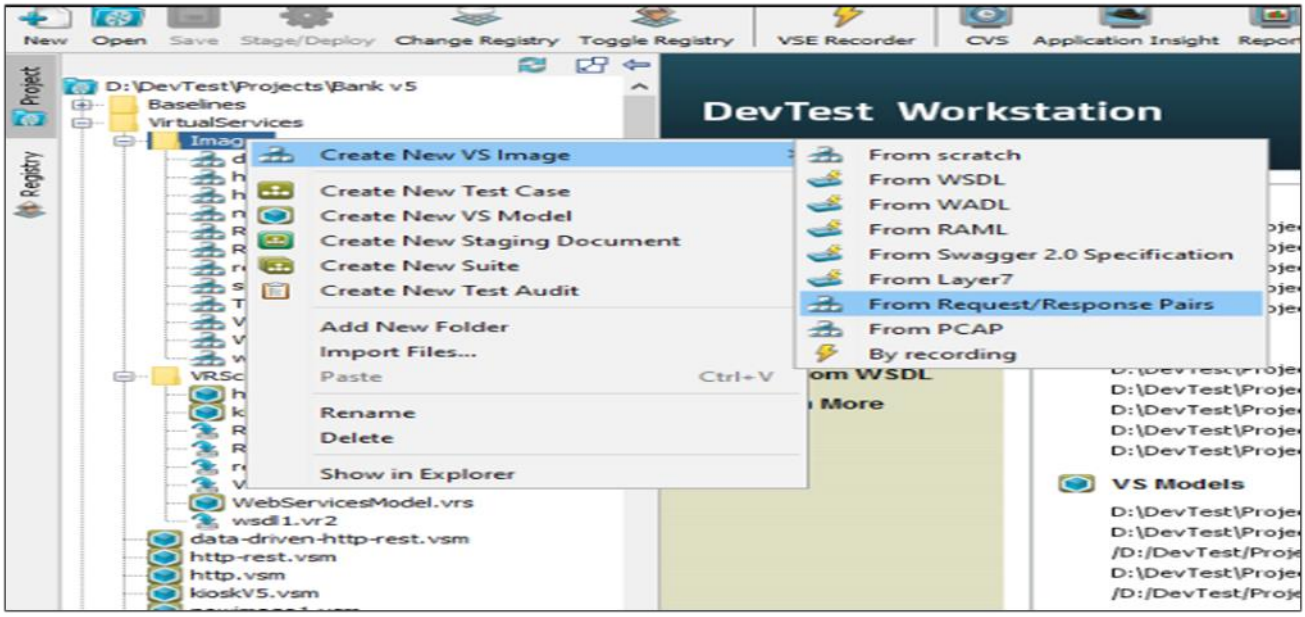
Notice that the month, day, and year properties have not been replaced with actual values. This result is expected, because there is no data set added to the test case.

23. JSON Scenarios

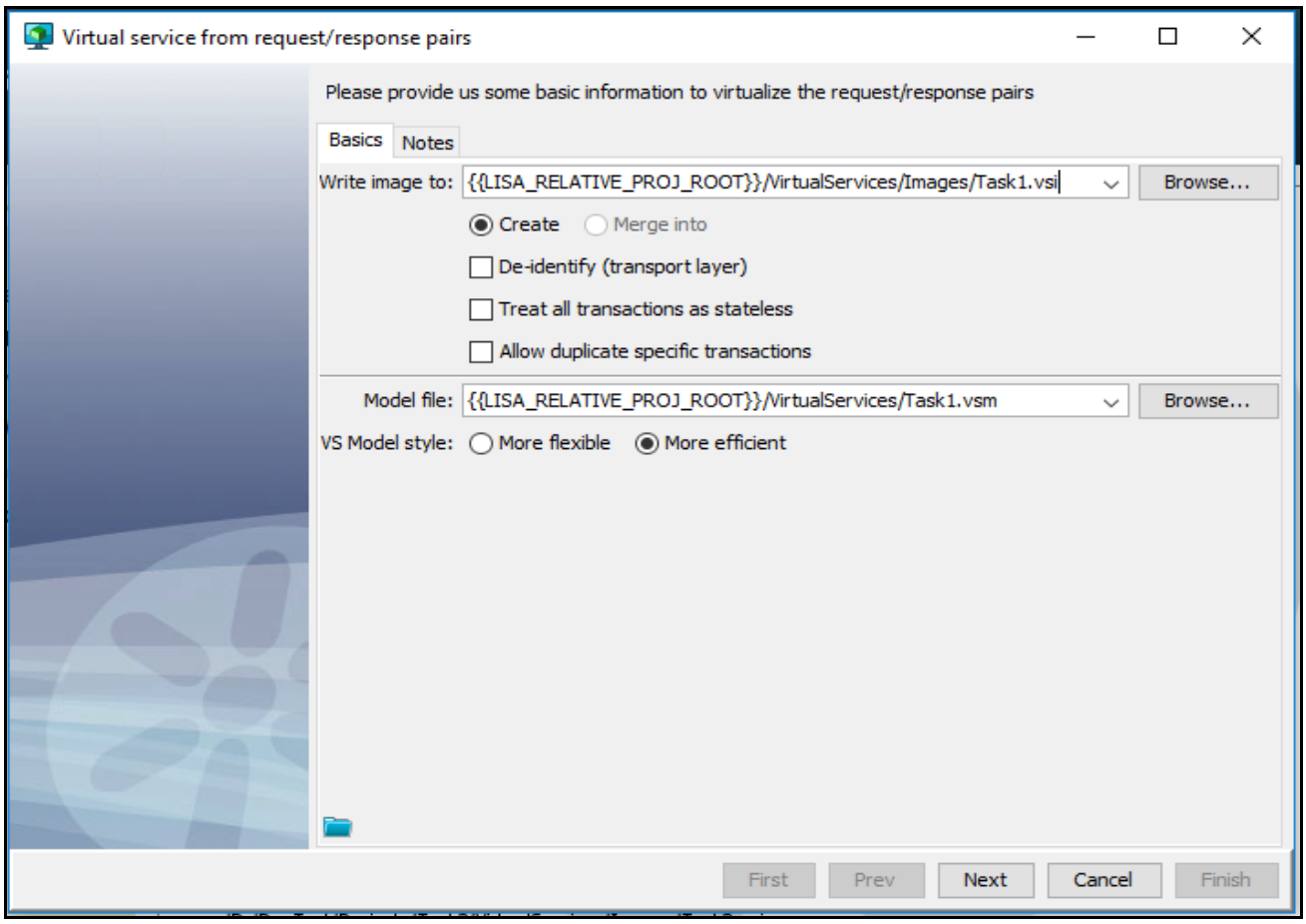
23.1 Virtualizing a Rest GET method with 10 unique transactions

Below are the steps for Virtualizing a Rest GET method with 10 unique transactions in terms of different input parameter values. If input parameter is not matched with 10 recorded transactions, then proper error message like “Input parameter is invalid” is displayed.

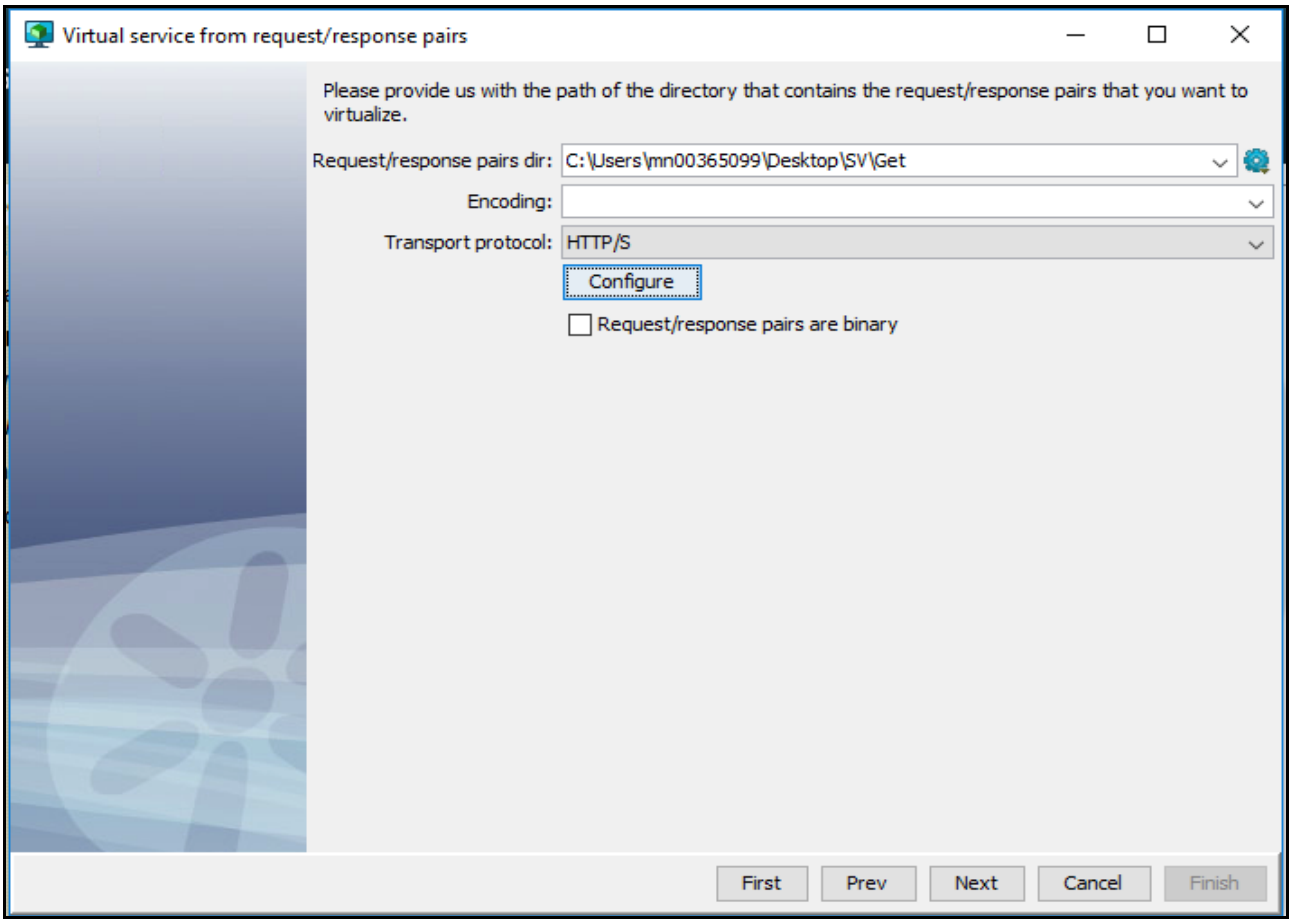
1. Select Projects > Images > Create New VS Image > From Request/Response Pairs



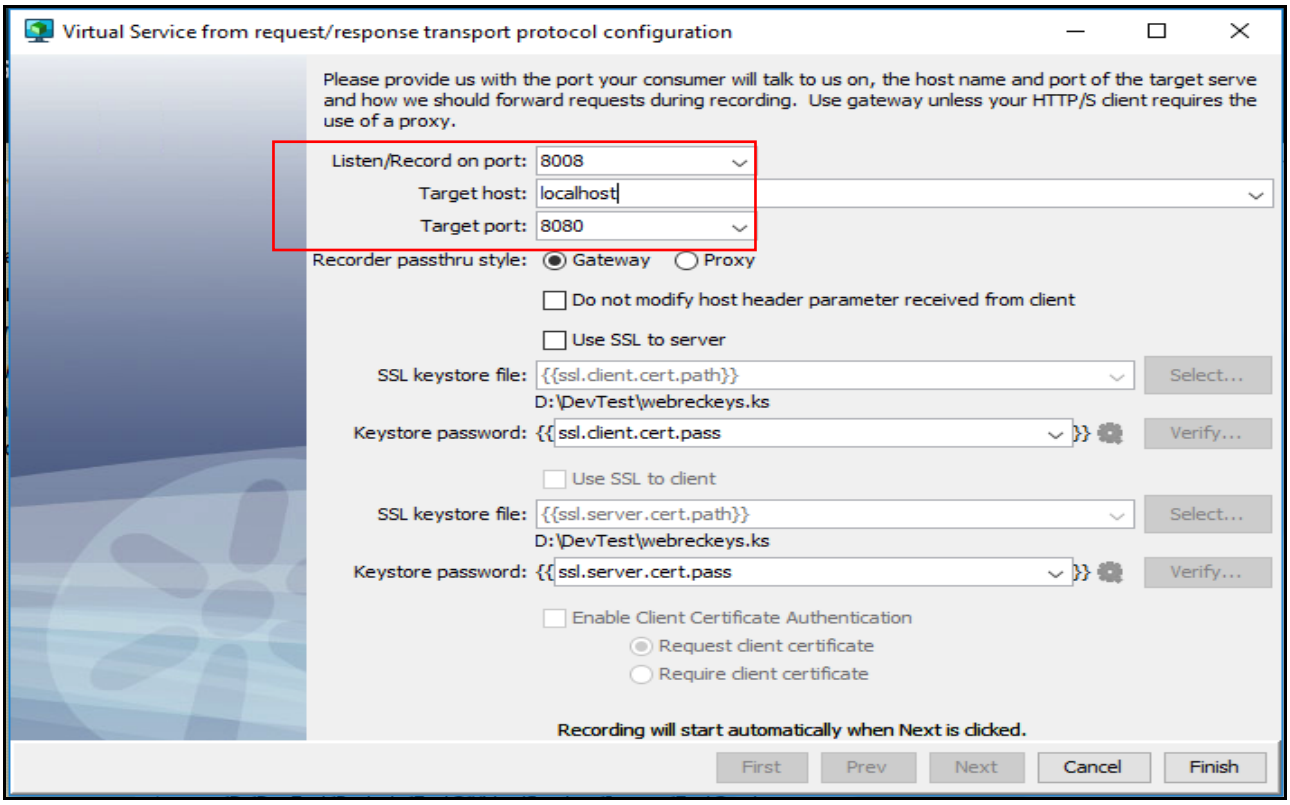
2. Enter Service Image name and Model name. VSI and VSM name should be same. Click **Next**.



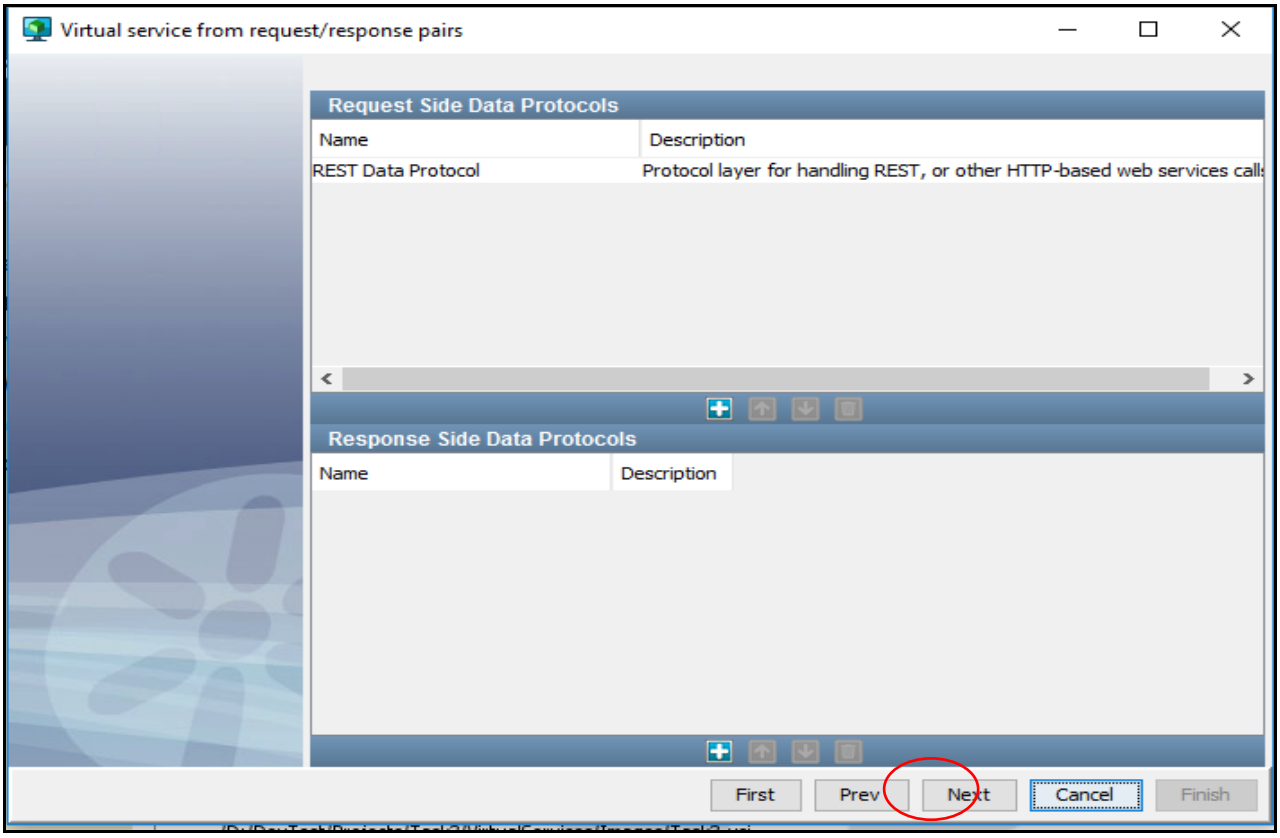
3. Browse the path for request/response pairs and select Transport protocol. Click **Configure**.



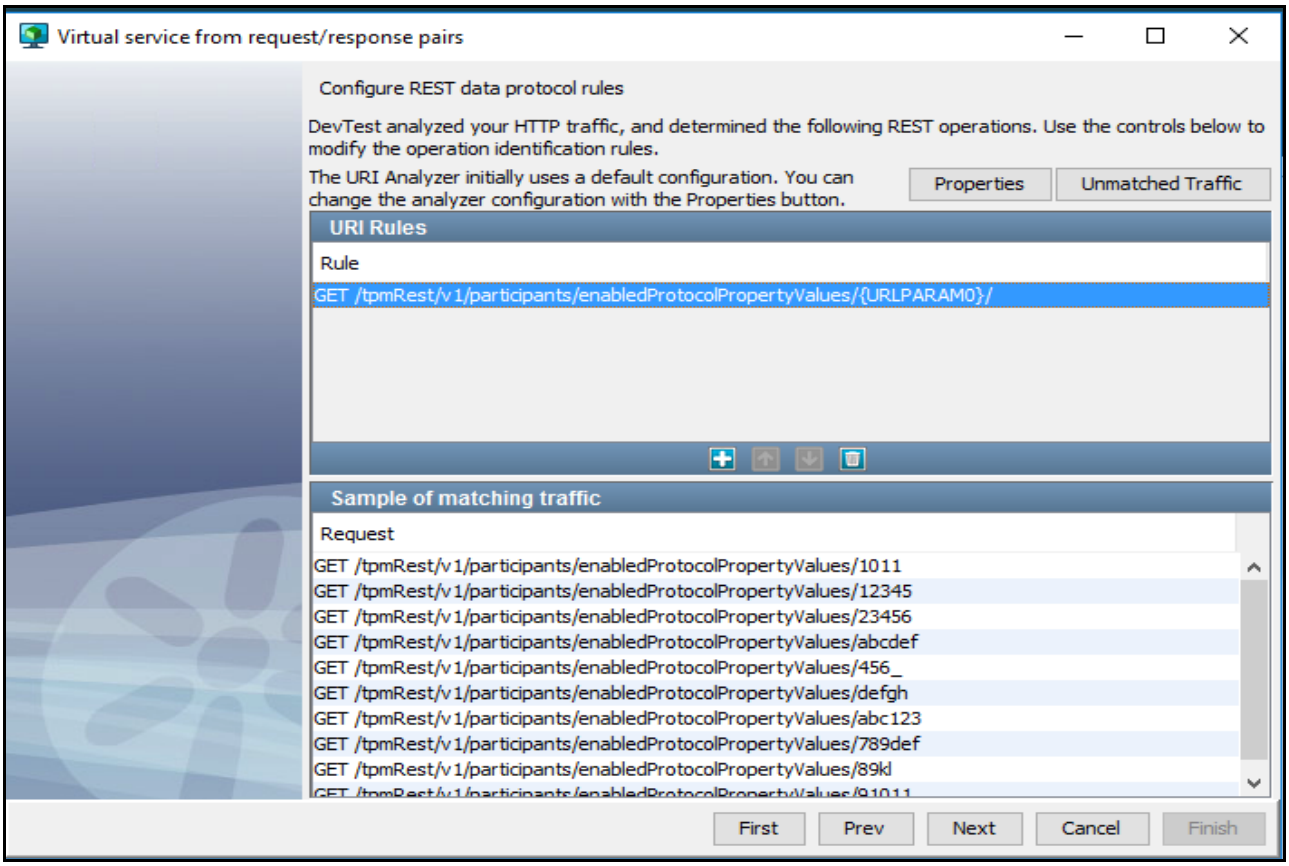
4. Provide necessary Listener port, localhost and target port and Click **Finish**.



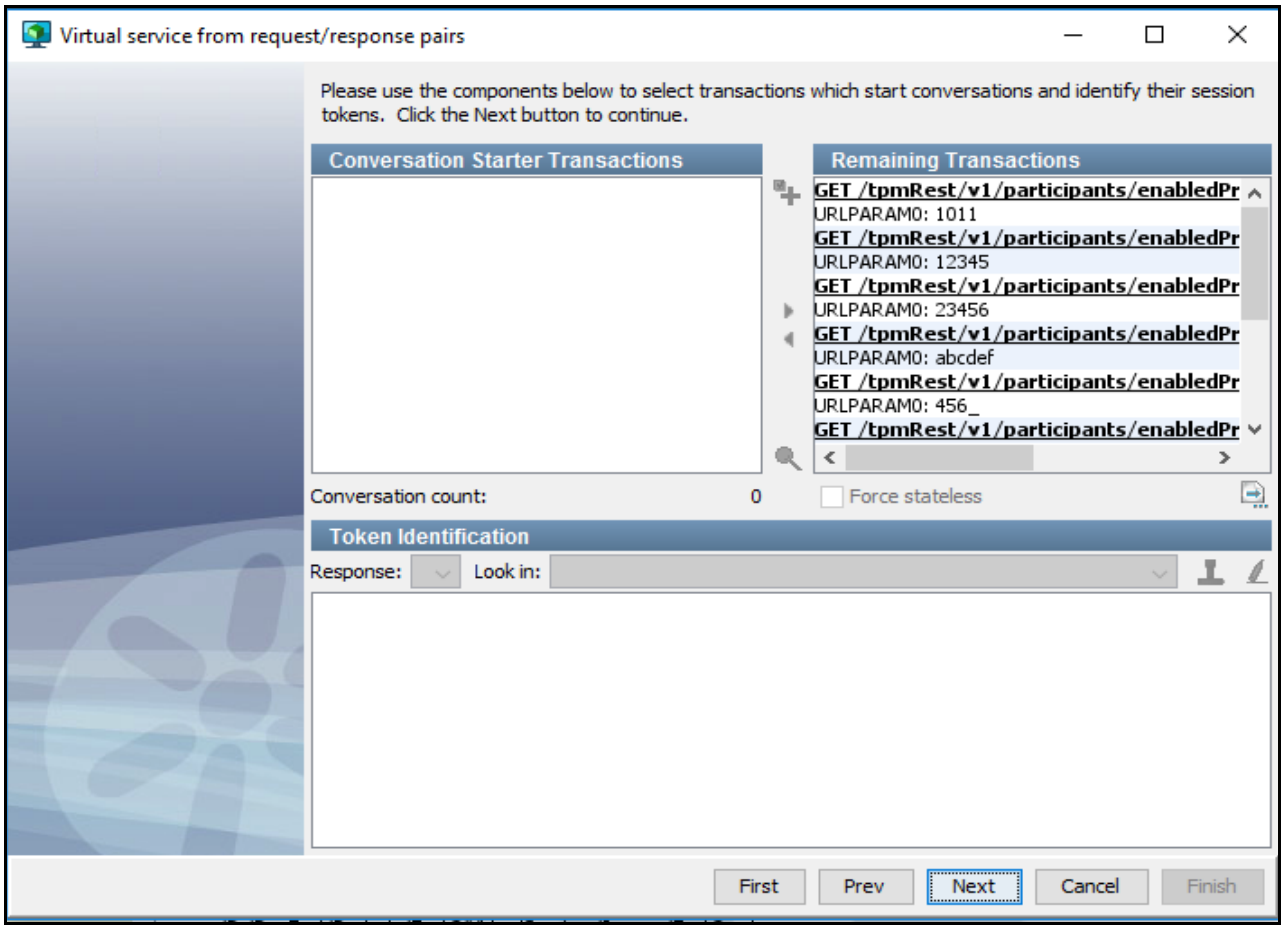
5. Appropriate data protocols for the request and response will be defaulted and click **Next**.



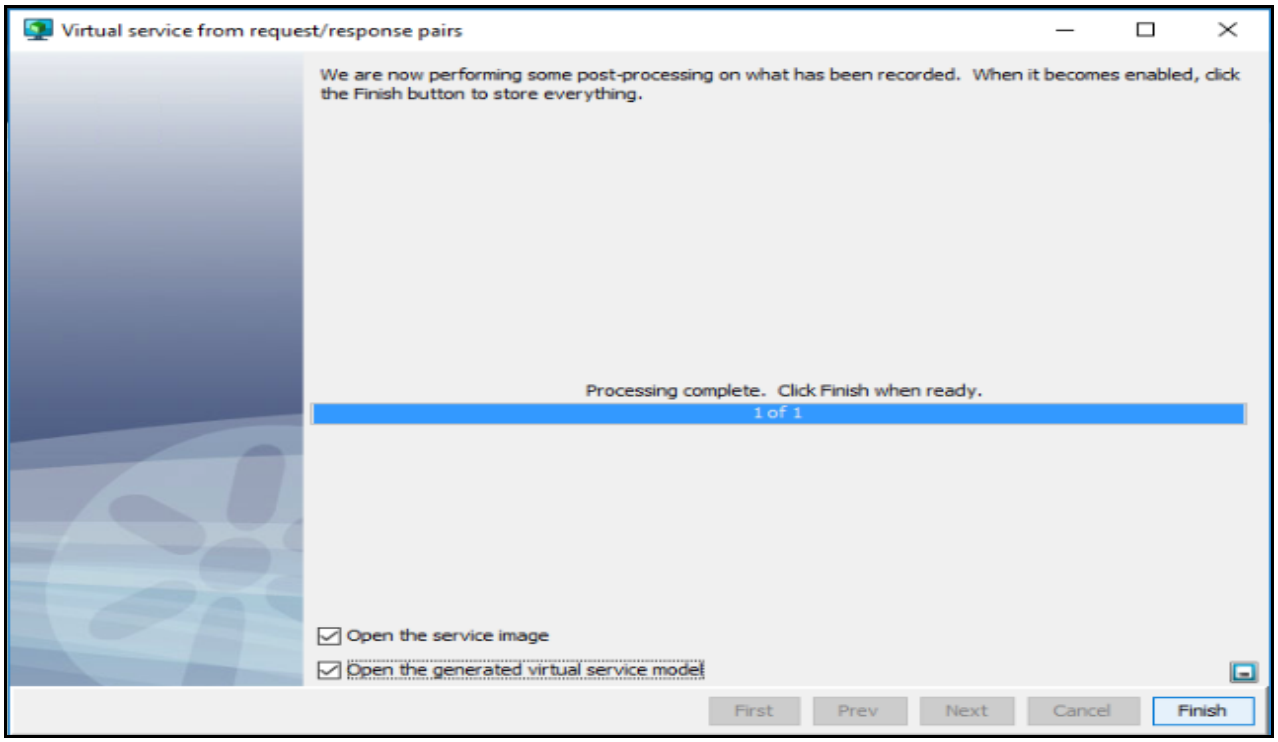
6. Appropriate URI rule will generated. Rule can be modified by going to properties.Click **Next.**



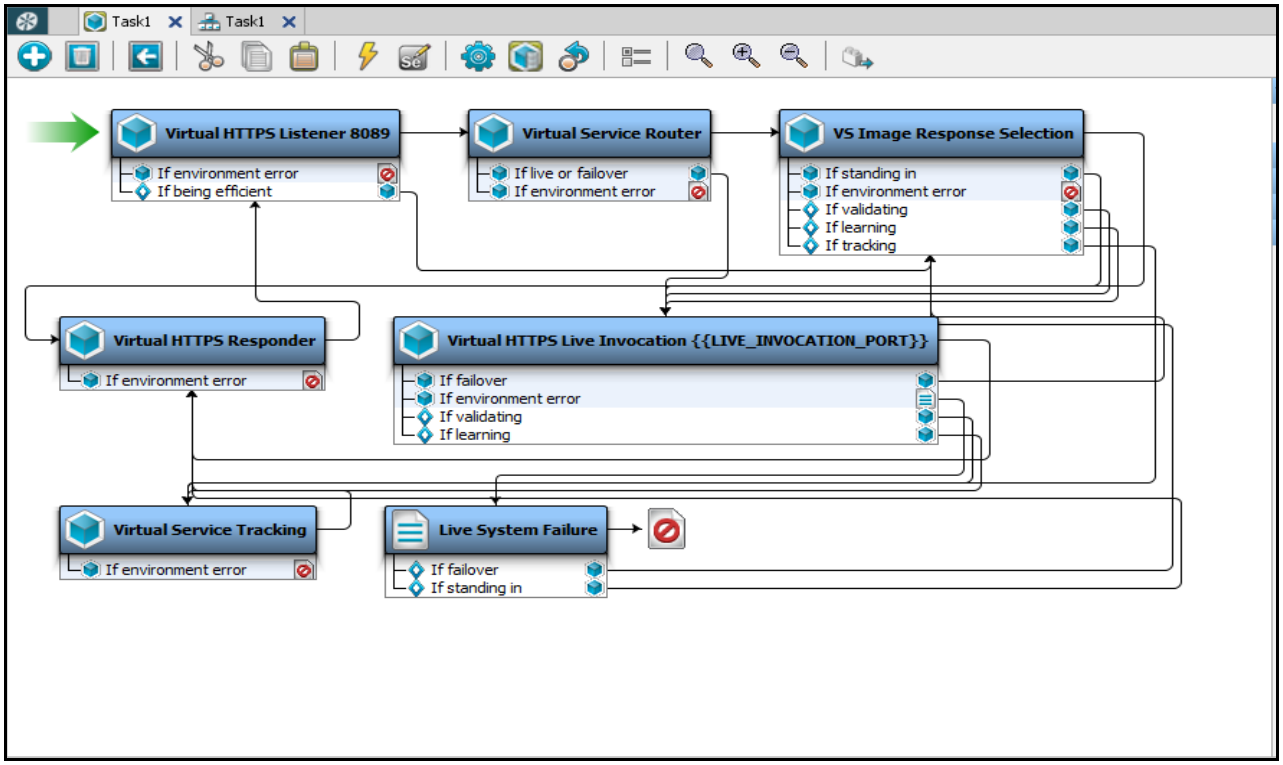
7. Click Next.


Virtual service from request/response pairs
Please use the components below to select transactions which start conversations and identify their session tokens. Click the Next button to continue.
Conversation Starter Transactions
Remaining Transactions
GET /tpmRest/v1/participants/enabledPr
URLPARAM0: 1011
GET /tpmRest/v1/participants/enabledPr
URLPARAM0: 12345
GET /tpmRest/v1/participants/enabledPr
URLPARAM0: 23456
GET /tpmRest/v1/participants/enabledPr
URLPARAM0: abcdef
GET /tpmRest/v1/participants/enabledPr
URLPARAM0: 456_
GET /tpmRest/v1/participants/enabledPr
Conversation count: 0
Force stateless
Token Identification
Response:
Look in:
First
Prev
Next
Cancel
Finish

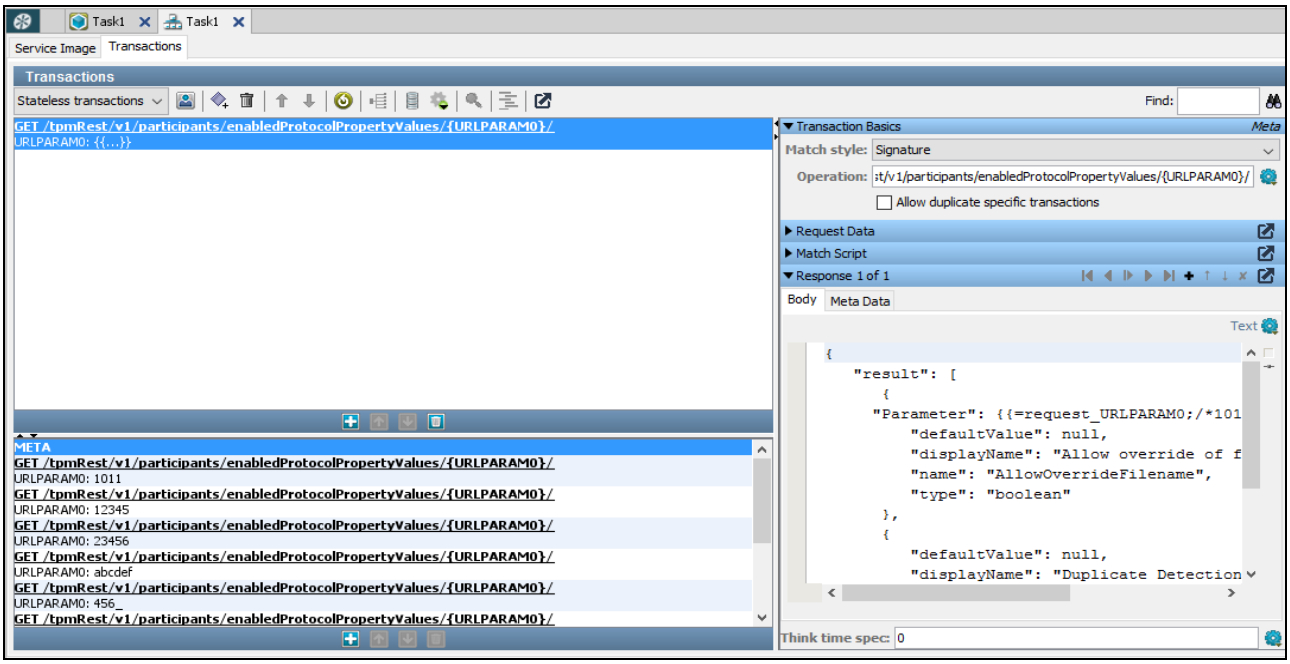
8. **Check** open VSI and VSM and click **finish**.



9. Virtual Service Model created.

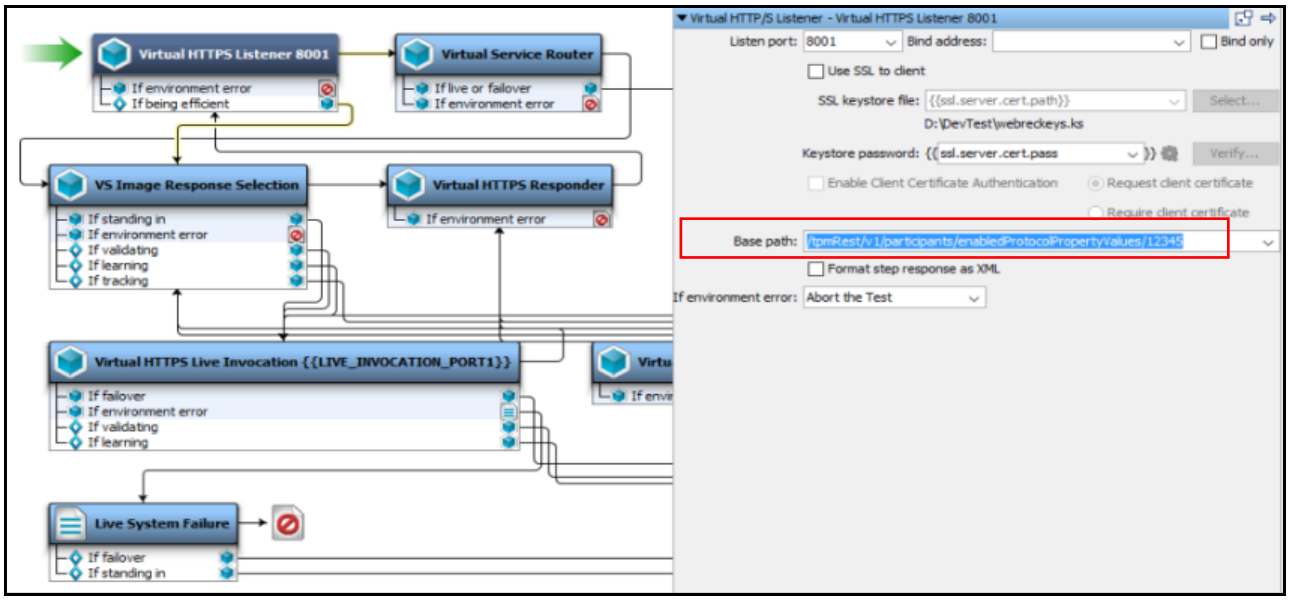


10. Virtual Service Image is created.

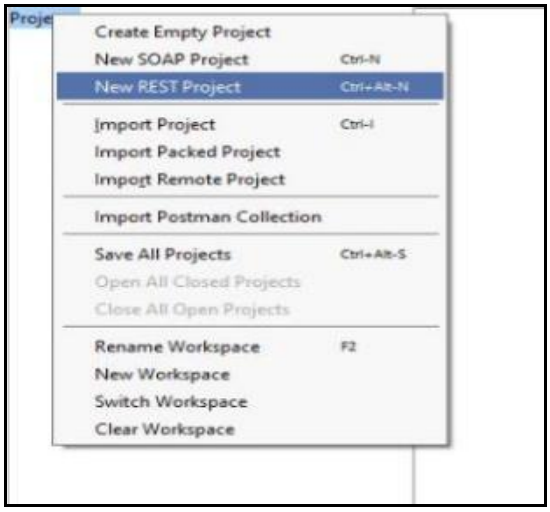


Now we will hit the same request from SOAP UI and test whether the response generated is same as in Virtual Service.

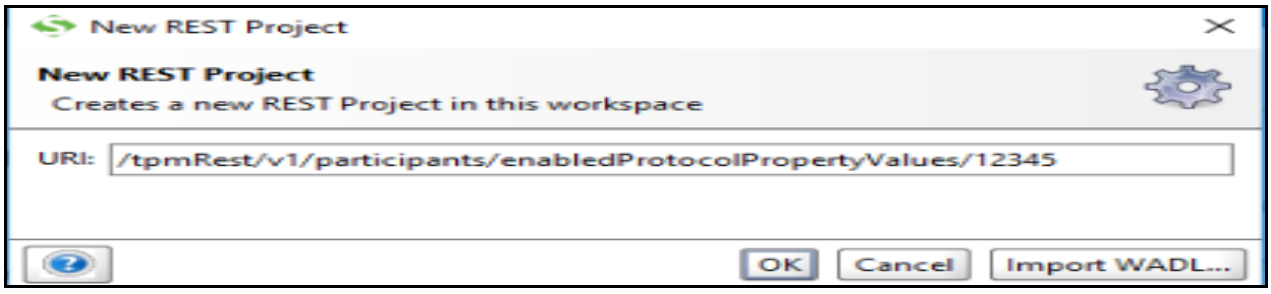
11. Double click the listener port and copy the base path.



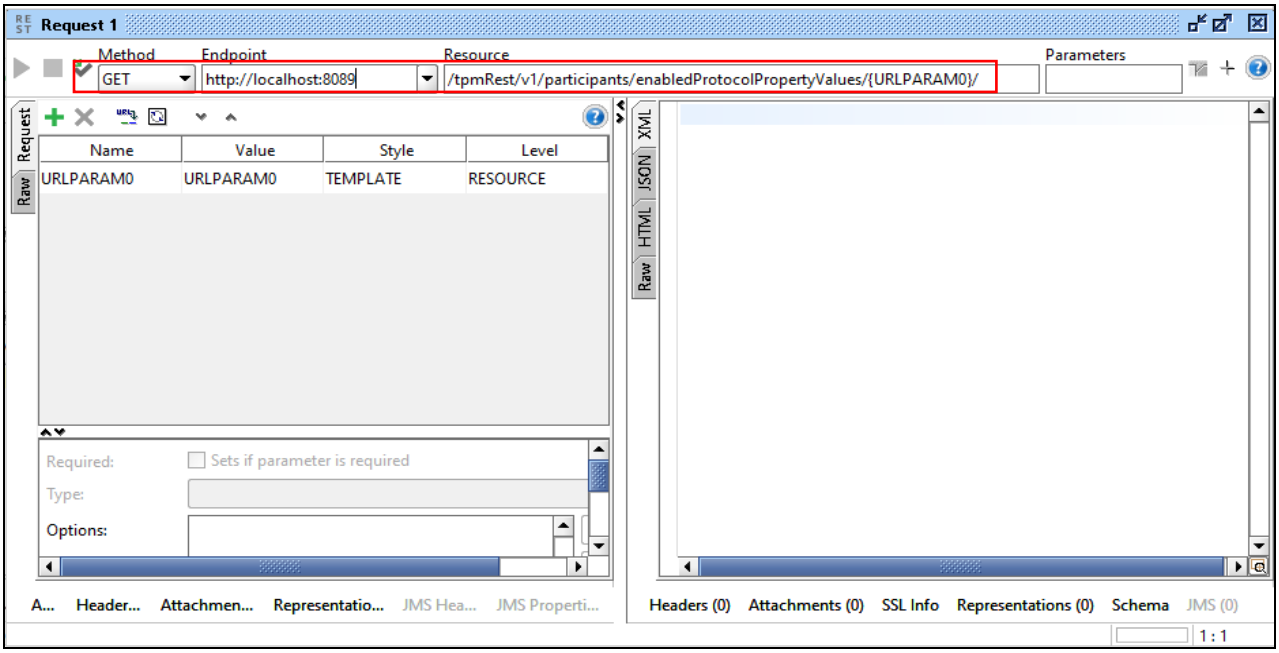
12. Go to SOAP UI > Projects > New Rest Project



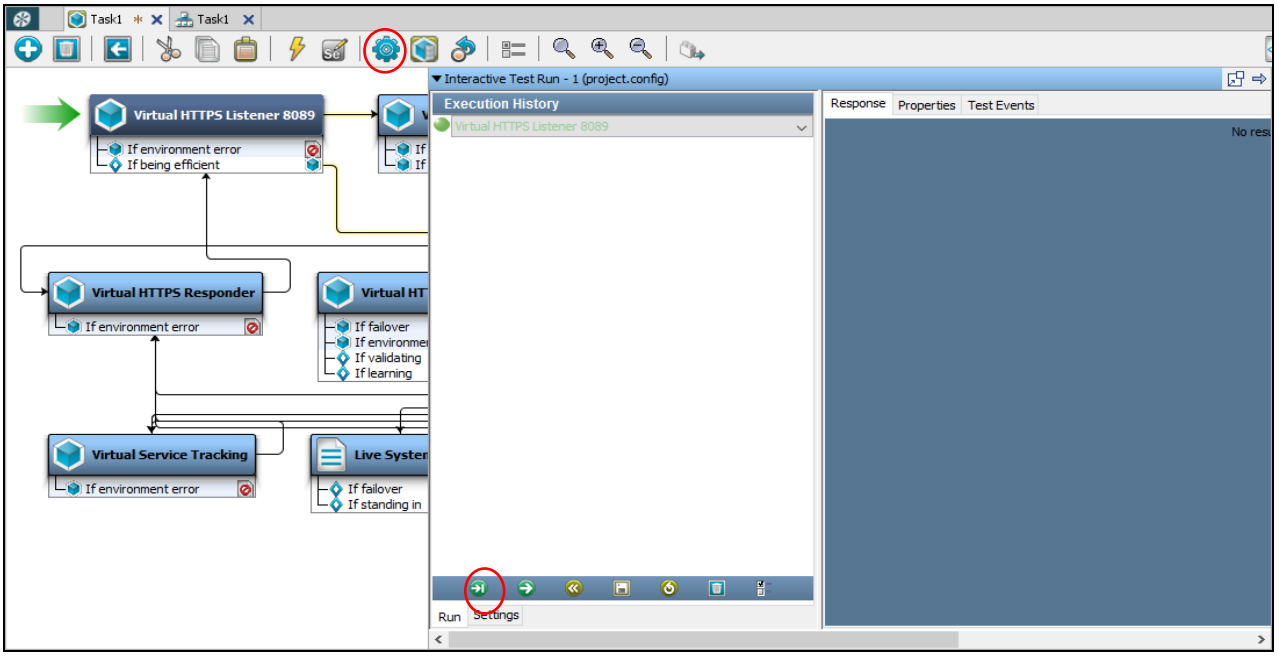
13. Enter the copied base path from VSM and click **Ok**.



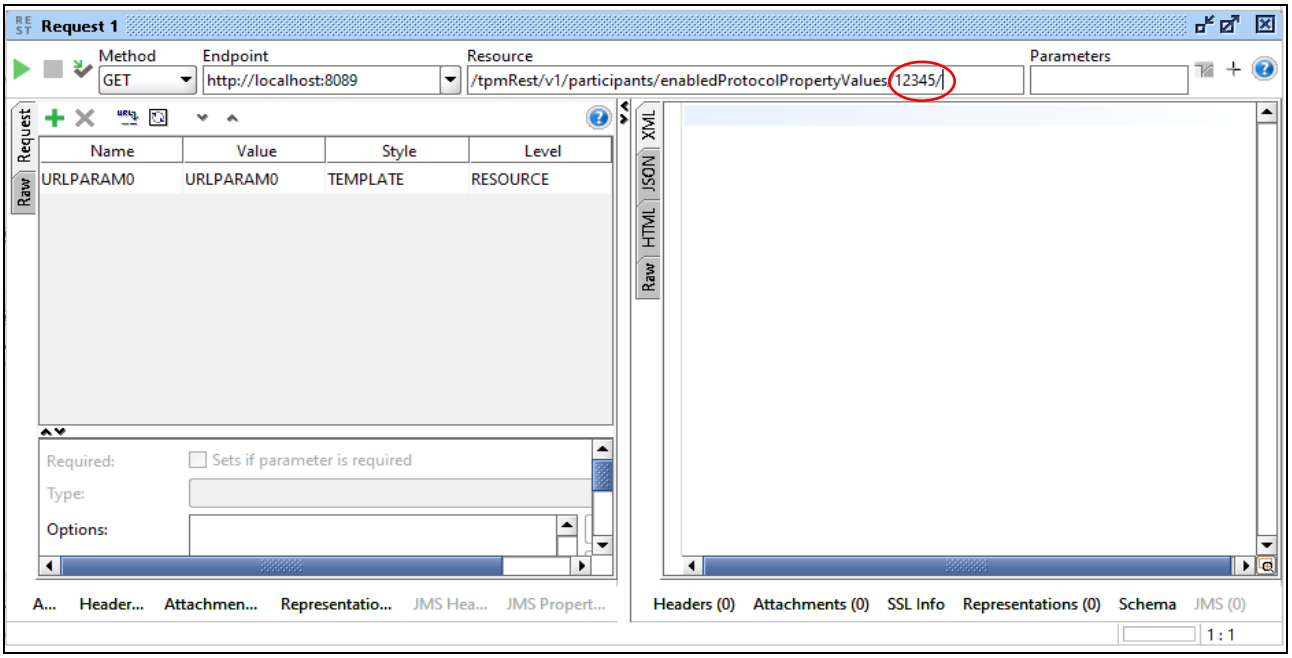
14. Enter the necessary endpoint, method, resource and parameters details
Endpoint: Transport protocol://target host: listener port.
Method: Get
Resource: Base Path



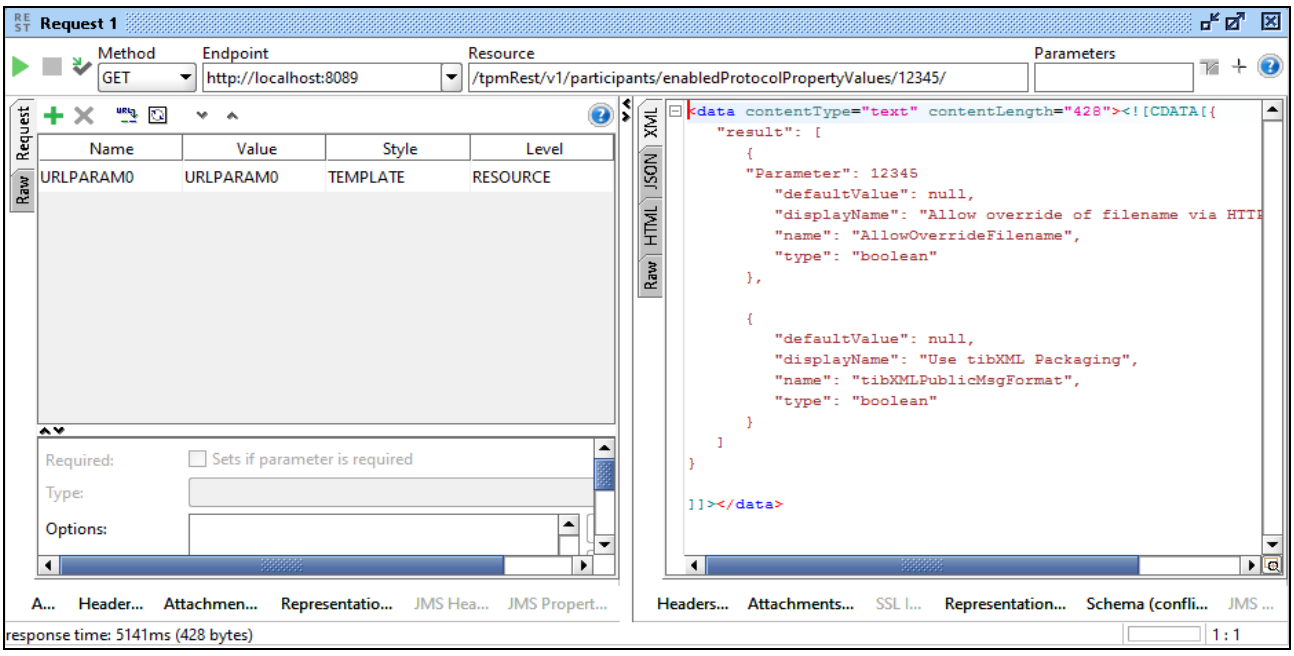
15. Go to VSM and start ITR and click on automatically execute the test.



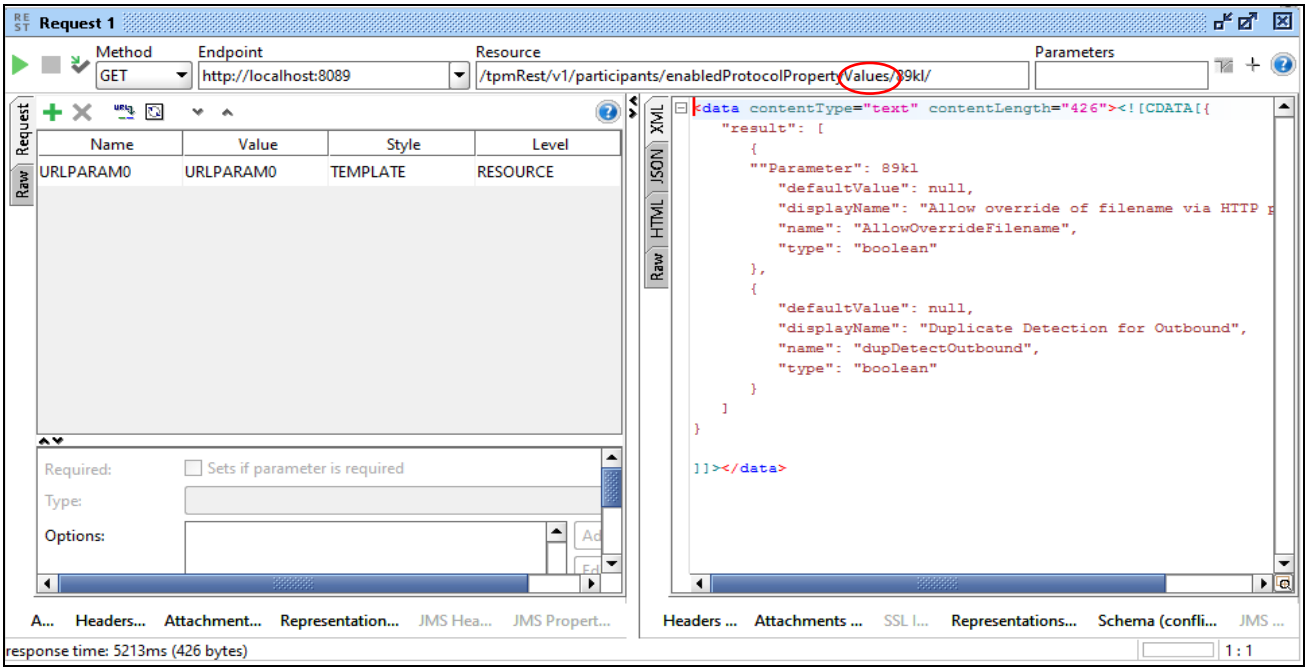
16. Give any of the input parameter that you have recorded in the 10 transactions.
E.g.: 12345



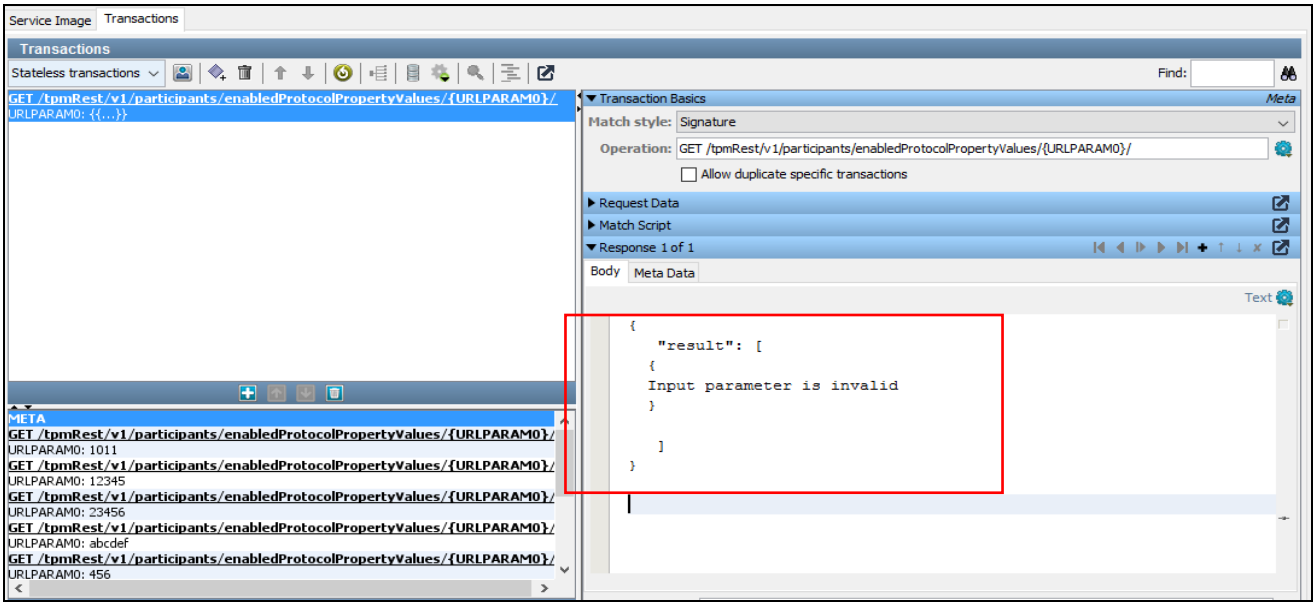
17. Click on run button in SOAP UI and it will hit the same response as in Virtual Service.



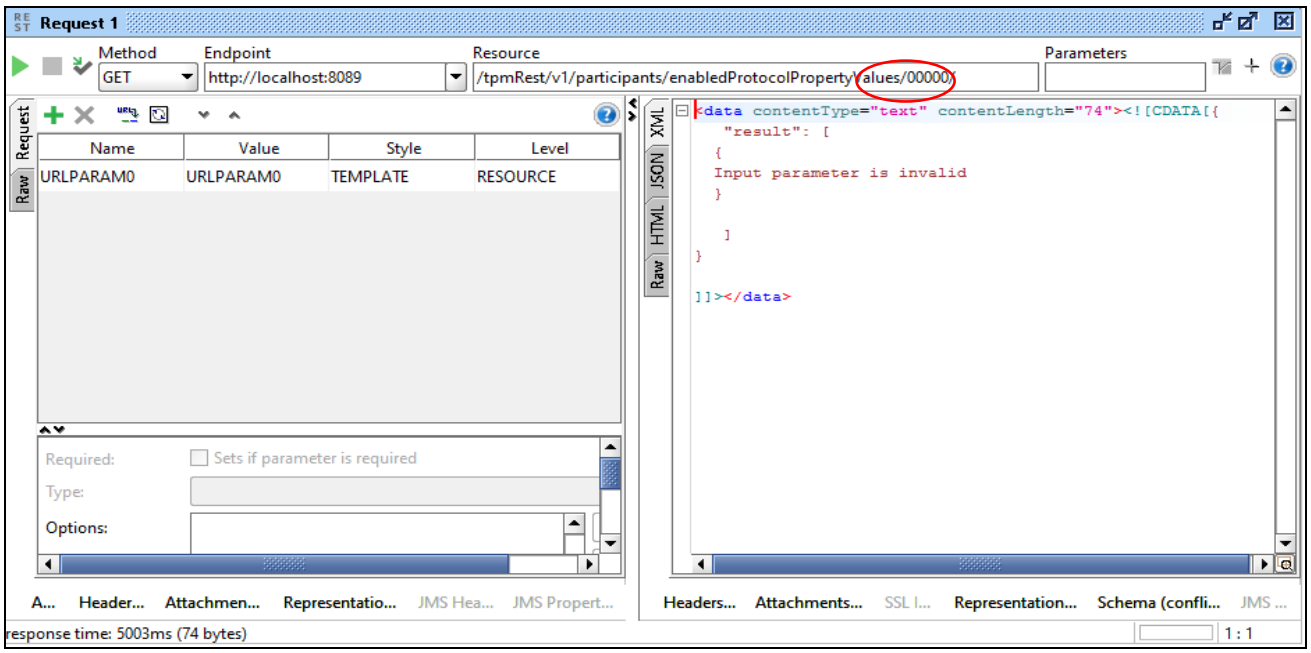
18. Another input parameter that you have recorded in the 10 transactions and run in SOAP UI.
E.g.: 89kl



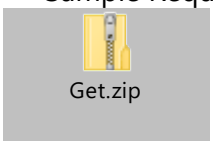
19. VSI > Meta > Response > Enter input parameter is invalid.



20. SOAP UI will hit the META Response if you give the input parameter that you have **NOT** recorded in the 10 transactions. **E.g.: 00000**



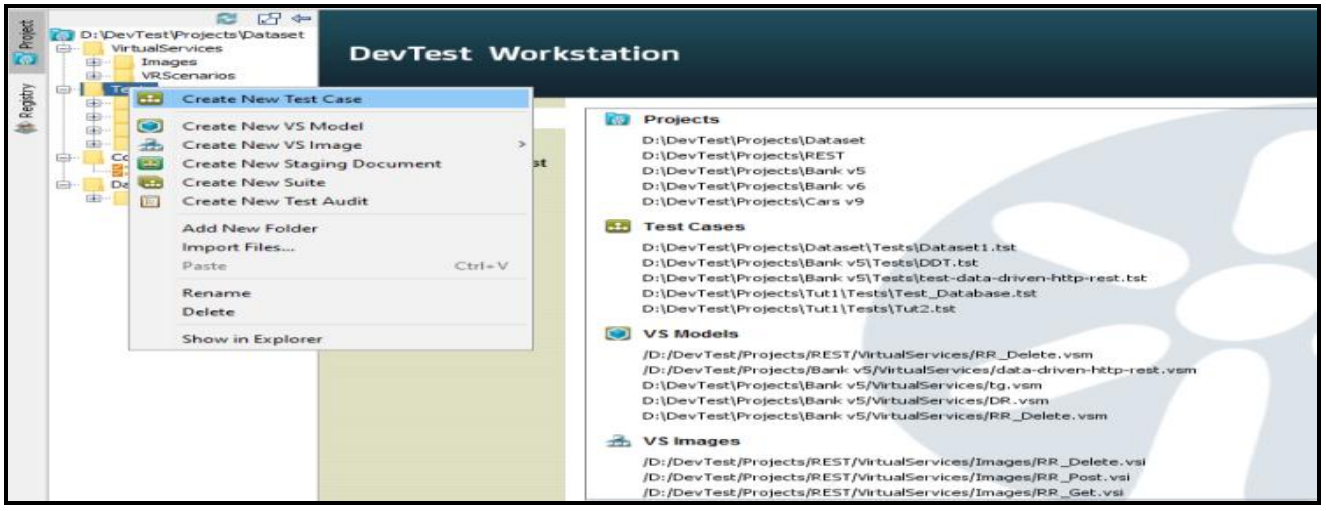
Sample Request and Response Files:



23.2 Inserting Unique Records in DB Table

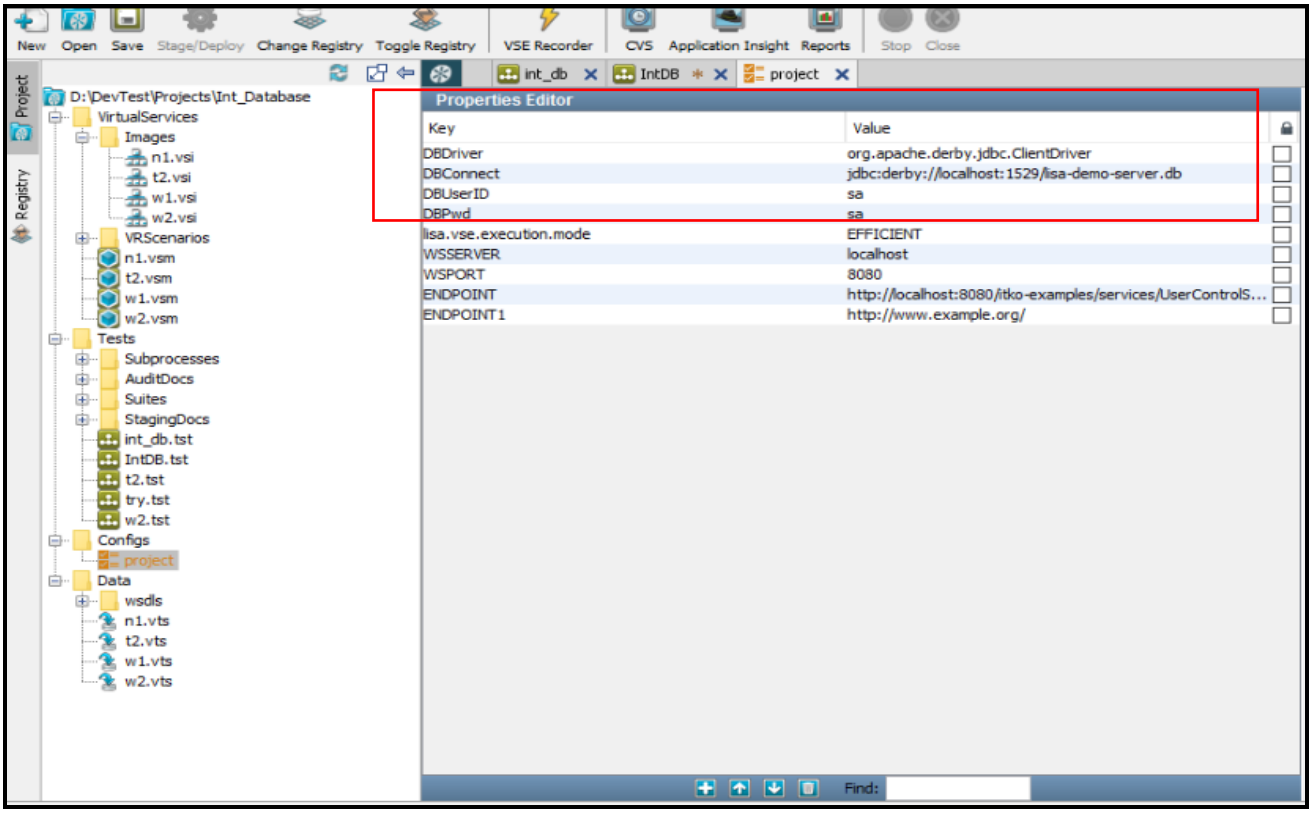
Inserting Unique Records in DB Table

1) Projects > Test case > Create New Test Case

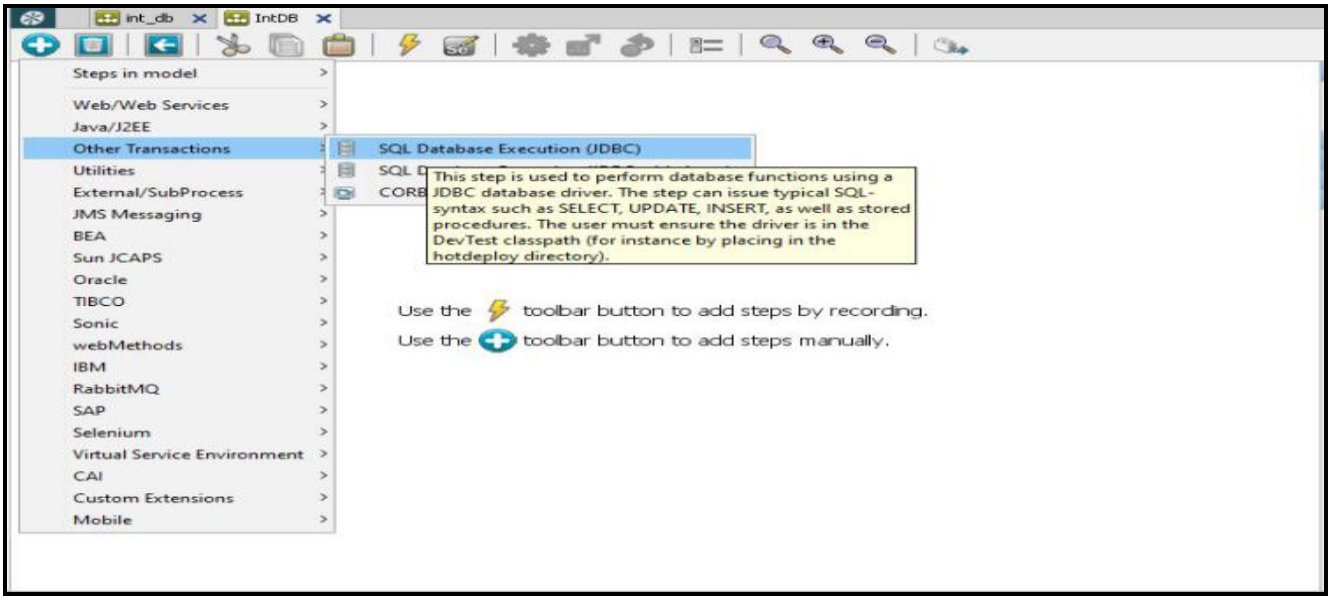


2) Configs > projects > Add following Database properties to the configuration.

- DBDriver: org.apache.derby.jdbc.ClientDriver
- DBConnect: jdbc:derby://localhost:1529/lisa-demo-server.db
- DBUserID: sa
- DBPwd: sa



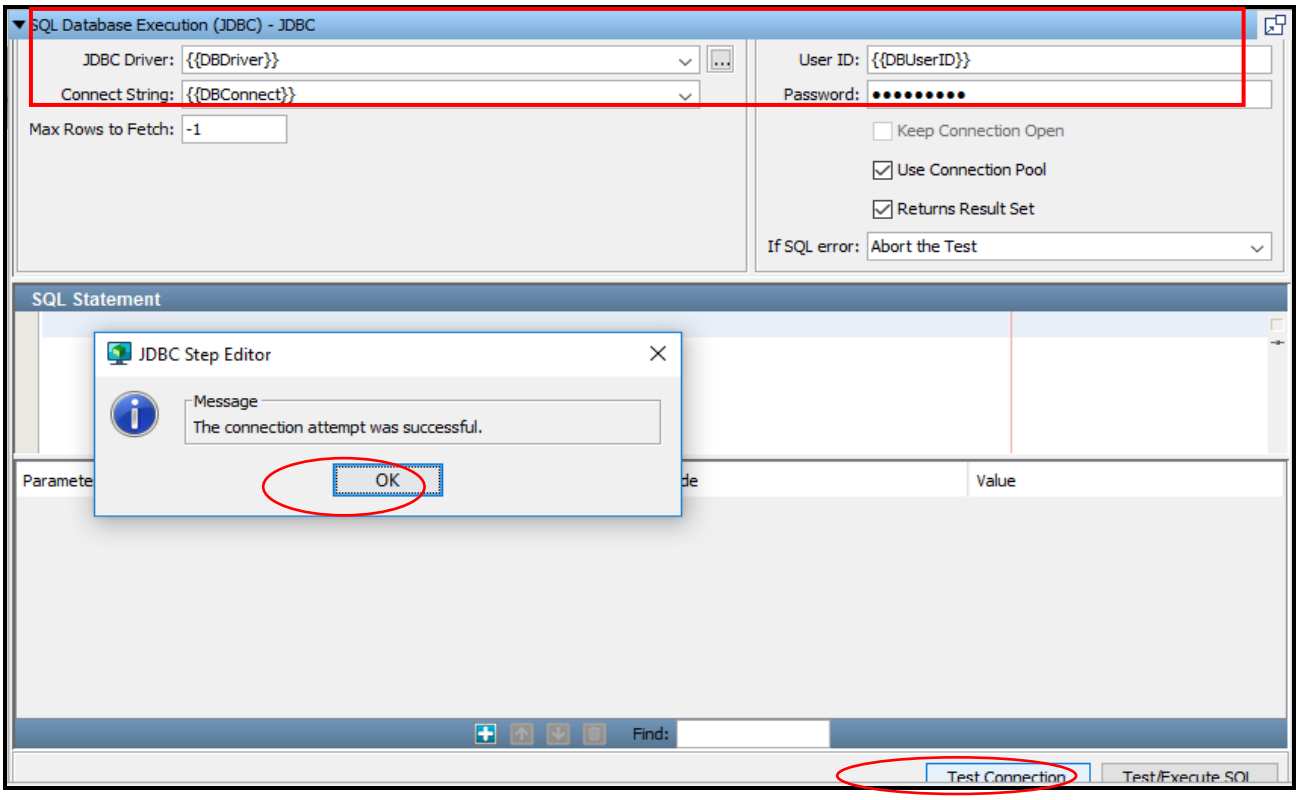
3) Click Add > Select Other transactions and SQL Database Execution(JDBC).



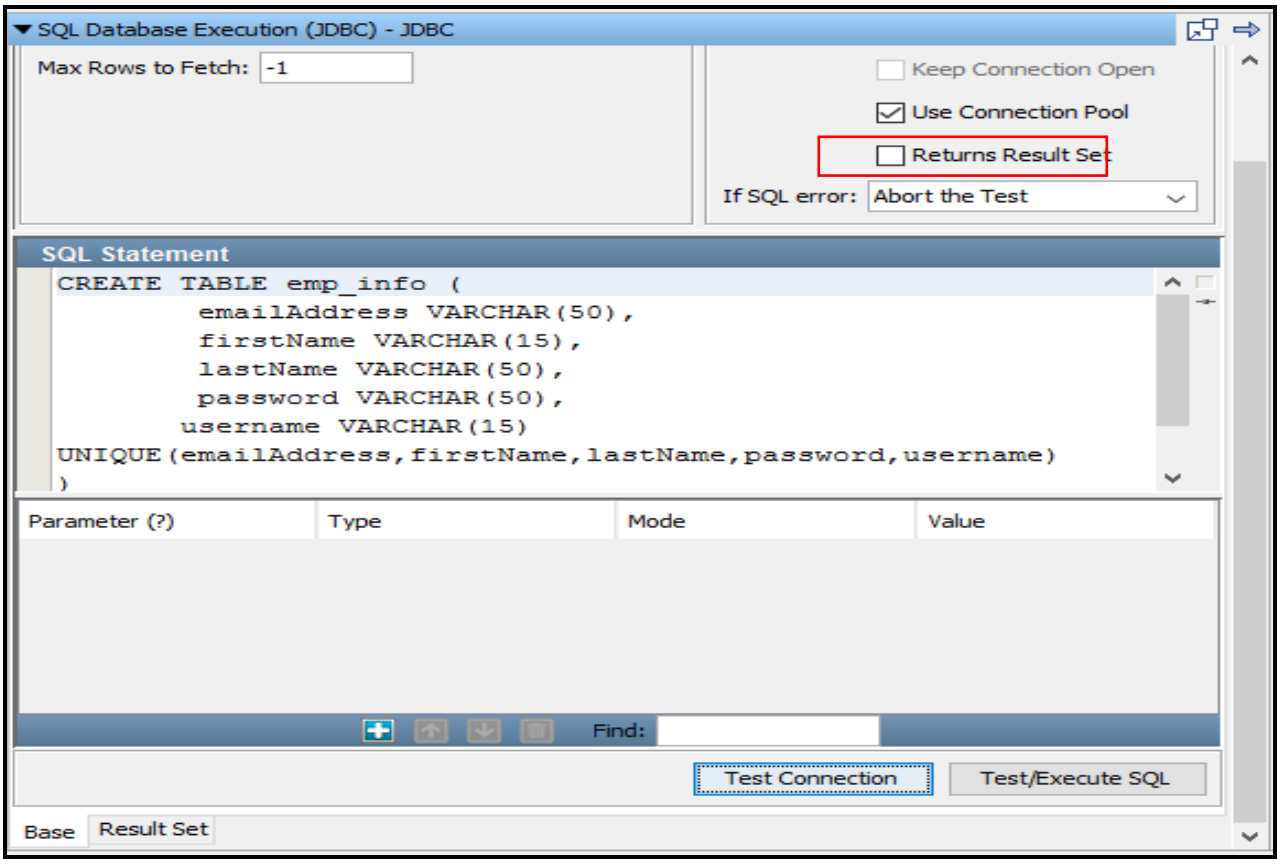
4) Double click on JDBC > enter the following to connect to database

- JDBC Driver: {{DBDriver}}
- Connect String: {{DBConnect}}
- User ID: {{DBUserID}}
- Password: {{DBPwd}}

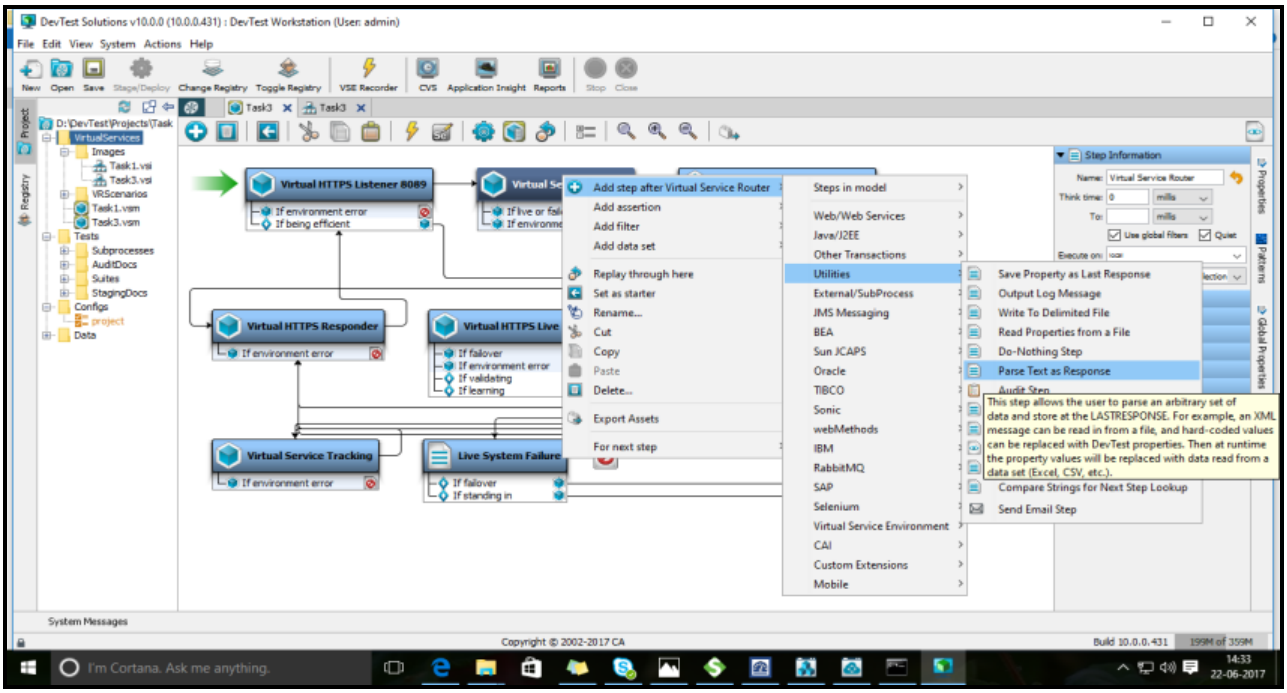
Click on Test connection



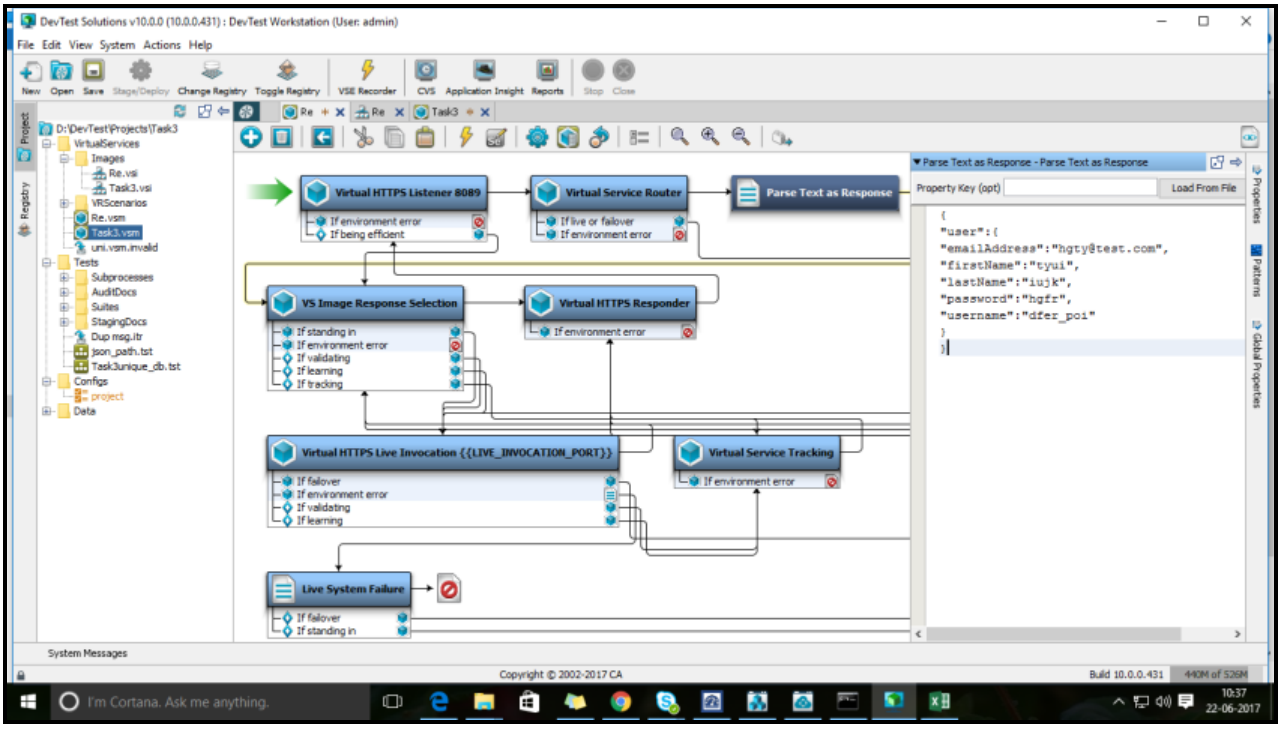
5) Create a Table. While creating uncheck the Return Result set. Click on Test/Execute SQL.



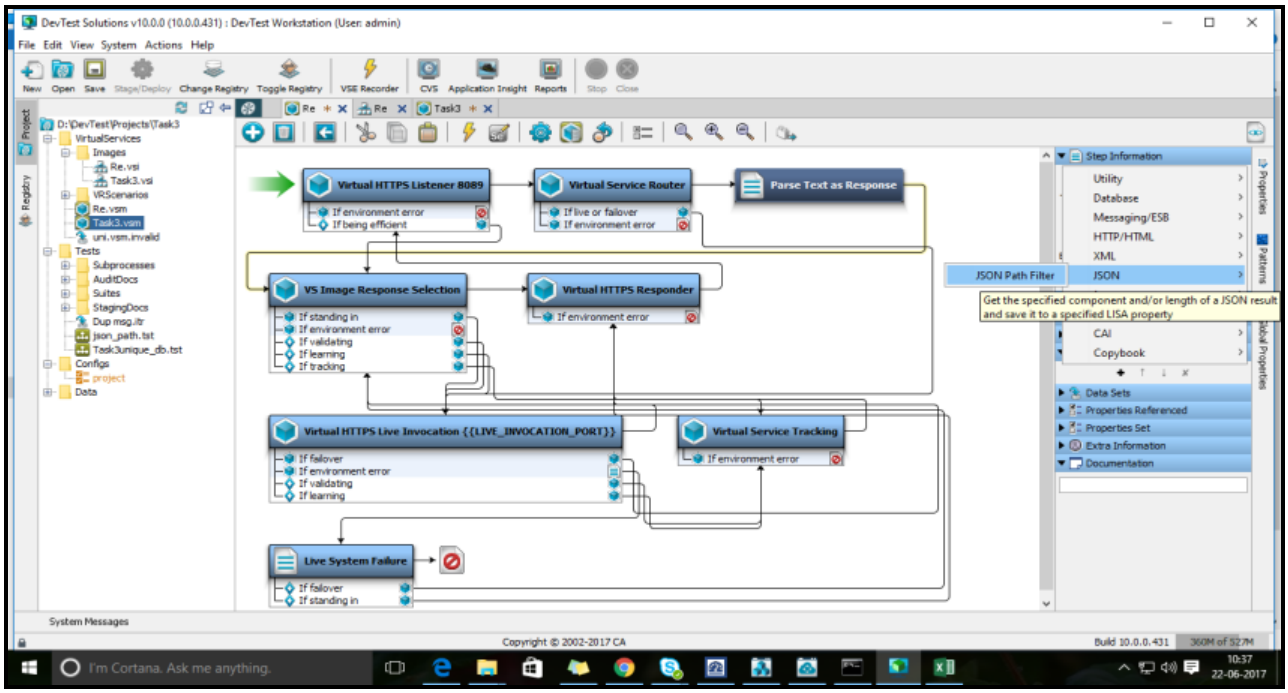
6) Create VSI and VSM using REST Service. After creating VSI & VSM, goto VSM and add Parse Text as Response step next to Virtual Service Execution Router.



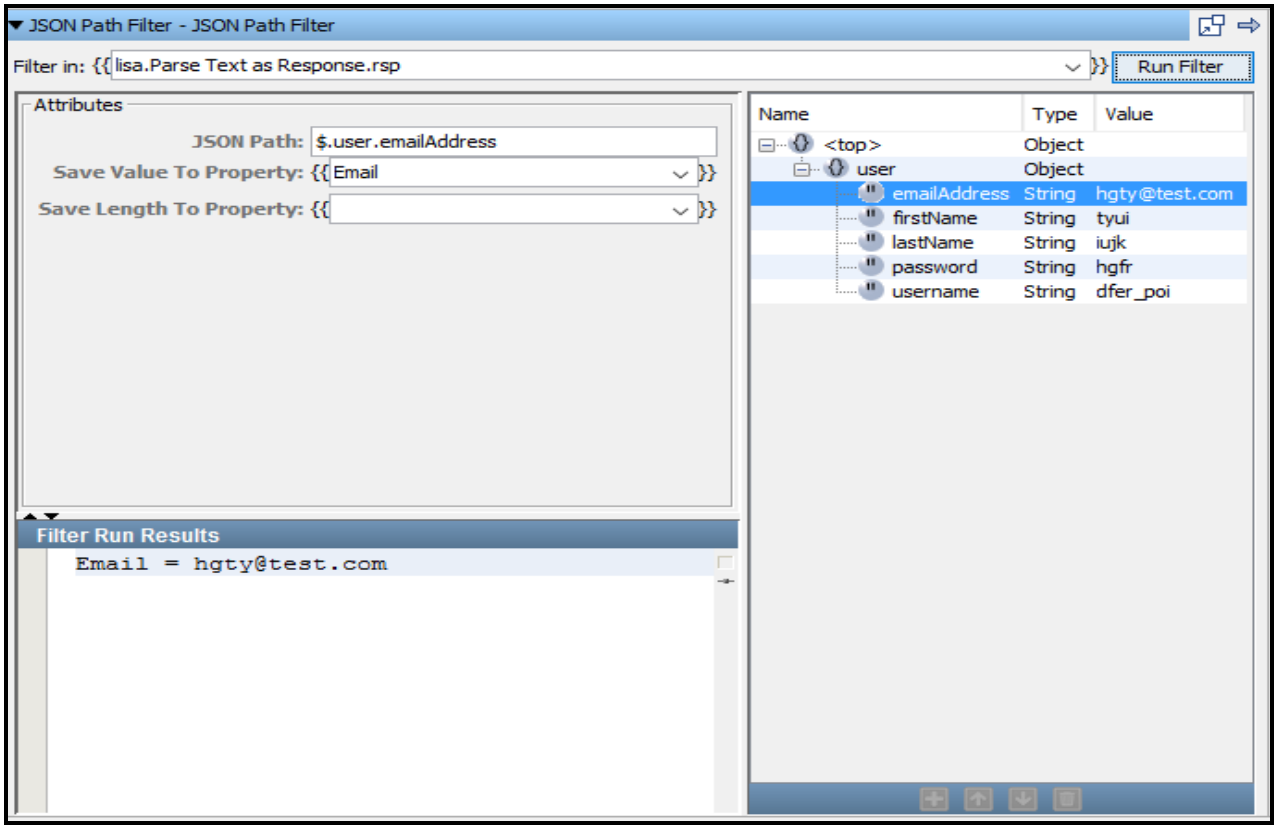
7) Click on Parse Text as Response step and add input parameters and test.

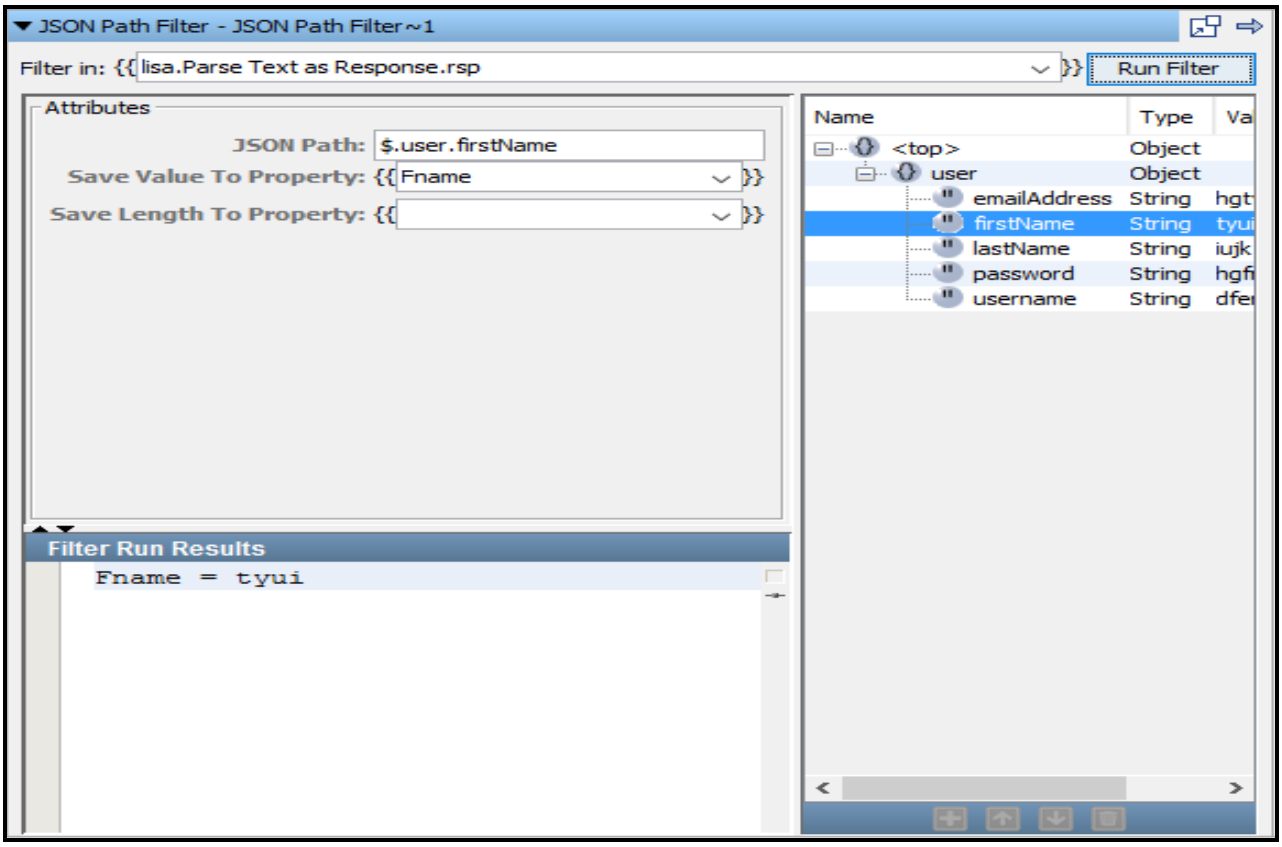


8) Double click the Parse Text as Response > Filters> **JSON Path Filter** to the step to store input parameter values.

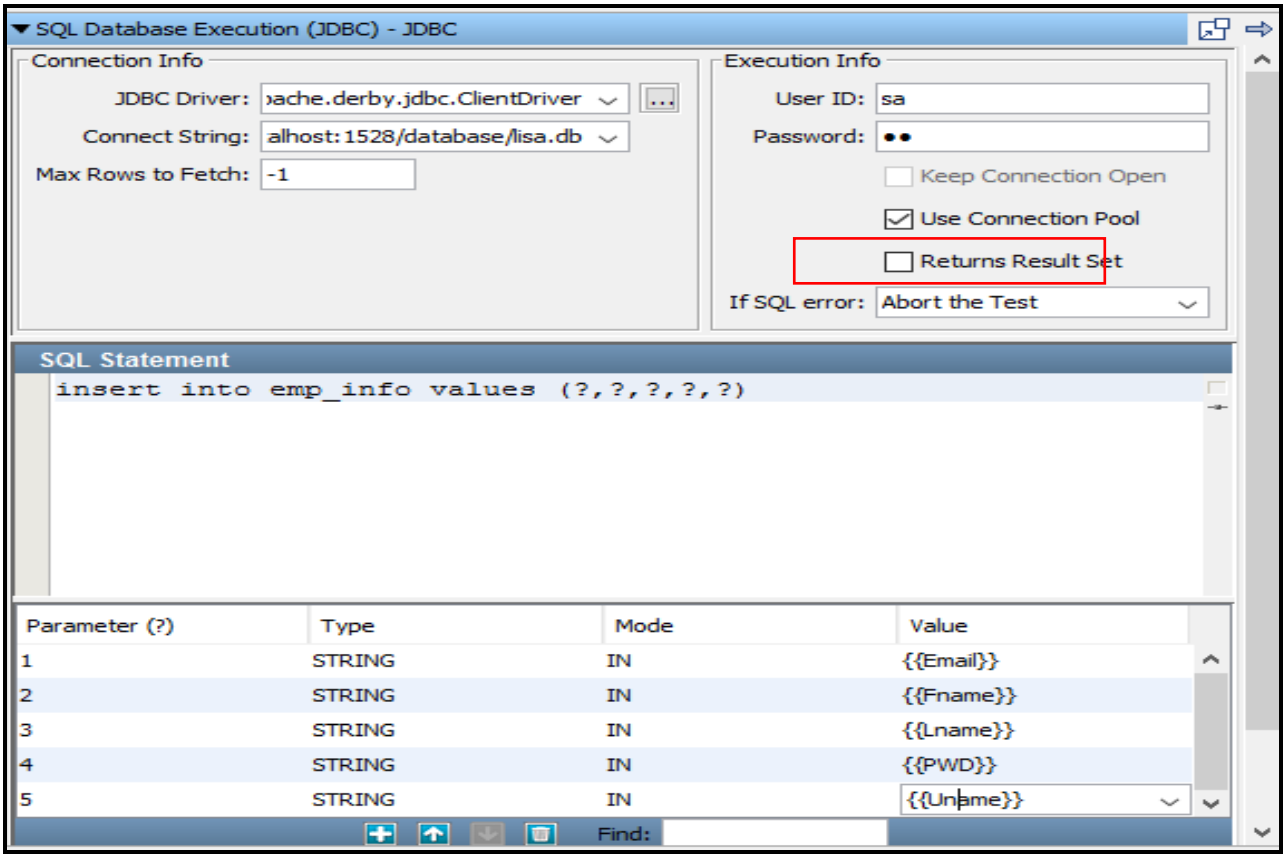


9) Create JSON Path Filters for all the input parameters

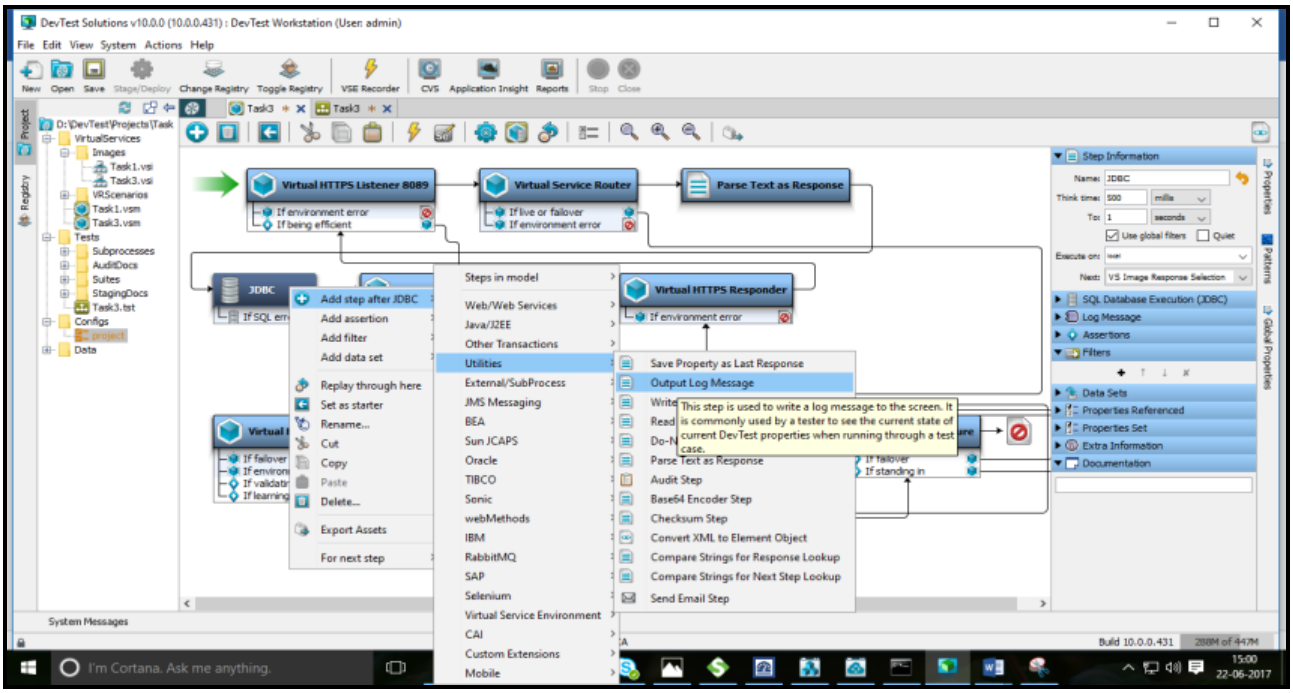



JSON Path Filter - JSON Path Filter~1
Filter in: {{ lisa.Parse Text as Response.rsp }}
Run Filter
Attributes
JSON Path: $.user.firstName
Save Value To Property: {{ Fname }}
Save Length To Property: {{ }}
Filter Run Results
Fname = tyui
Name
Type
<top>
Object
user
Object
emailAddress
String
firstName
String
lastName
String
password
String
username
String

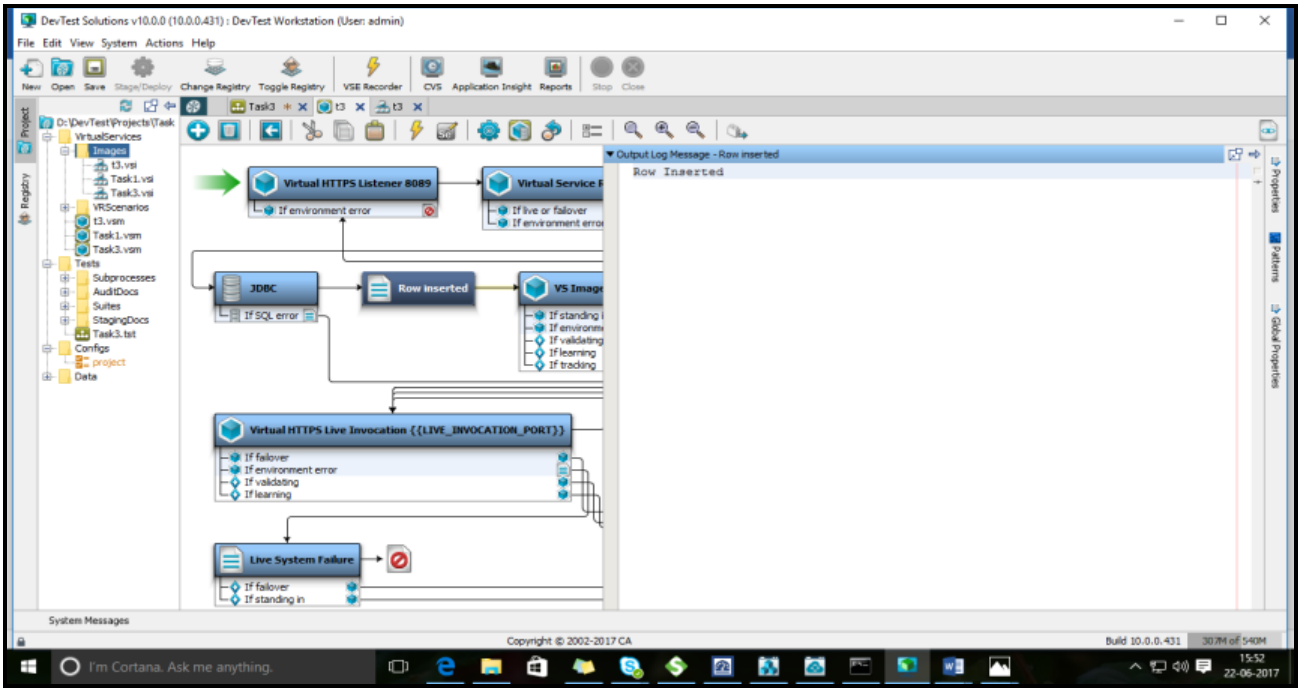
10) Copy and add **SQL Database Execution (JDBC) Test Step** which was created in step 3. Add **insert query** in the **SQL statement** w.r.t the table created. Also add the parameters that was created in JSON Path filter. Uncheck Returns Result Set



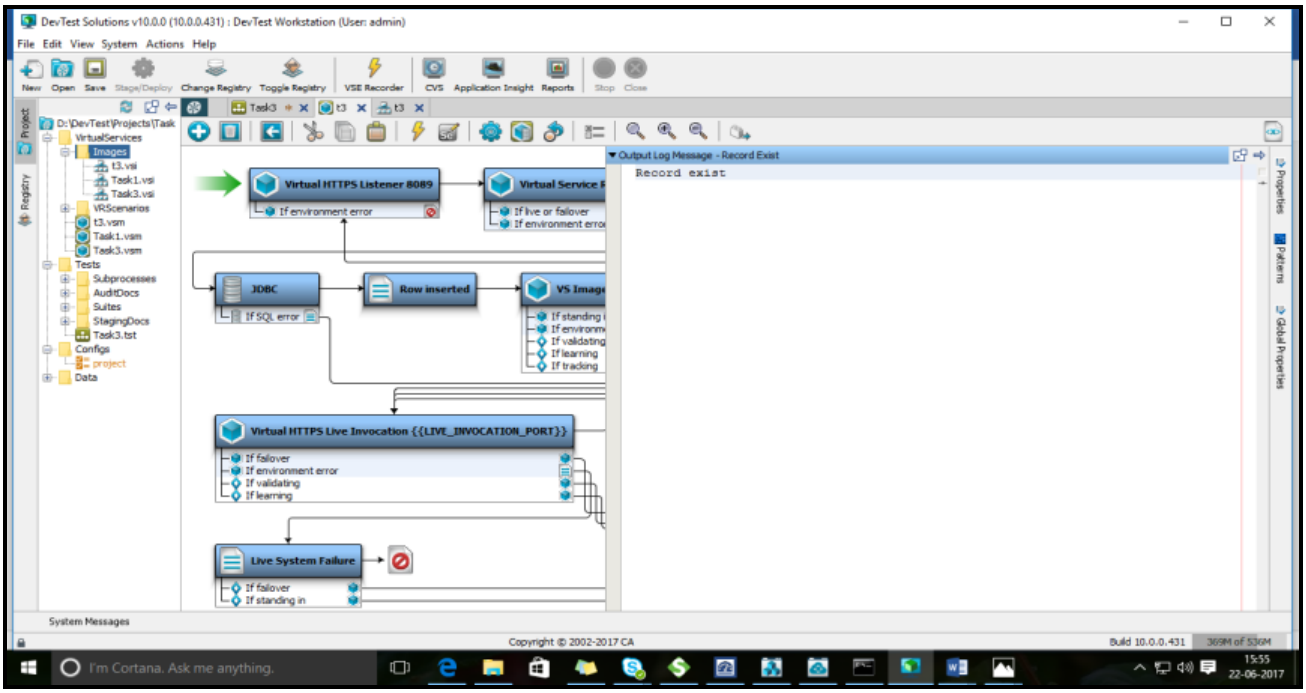
11) Add **Output Log Message** step after JDBC step.



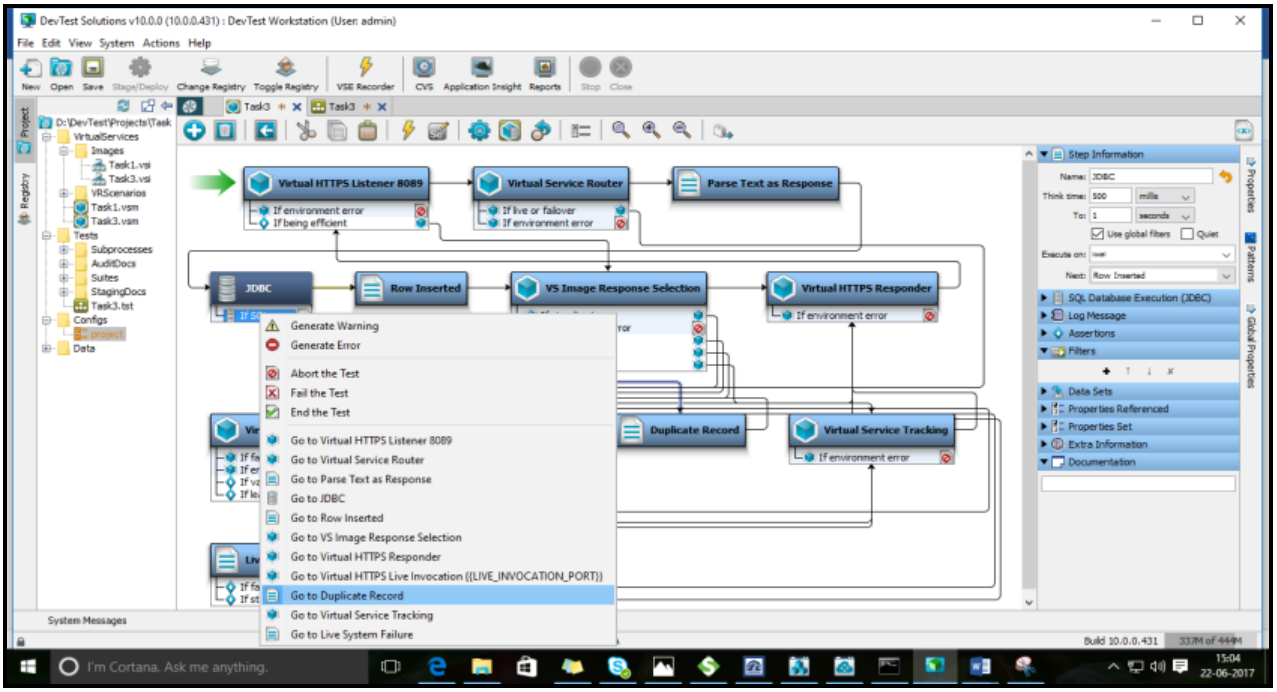
12) Double click Output Log Message > rename and edit the content stating "**Row inserted**".



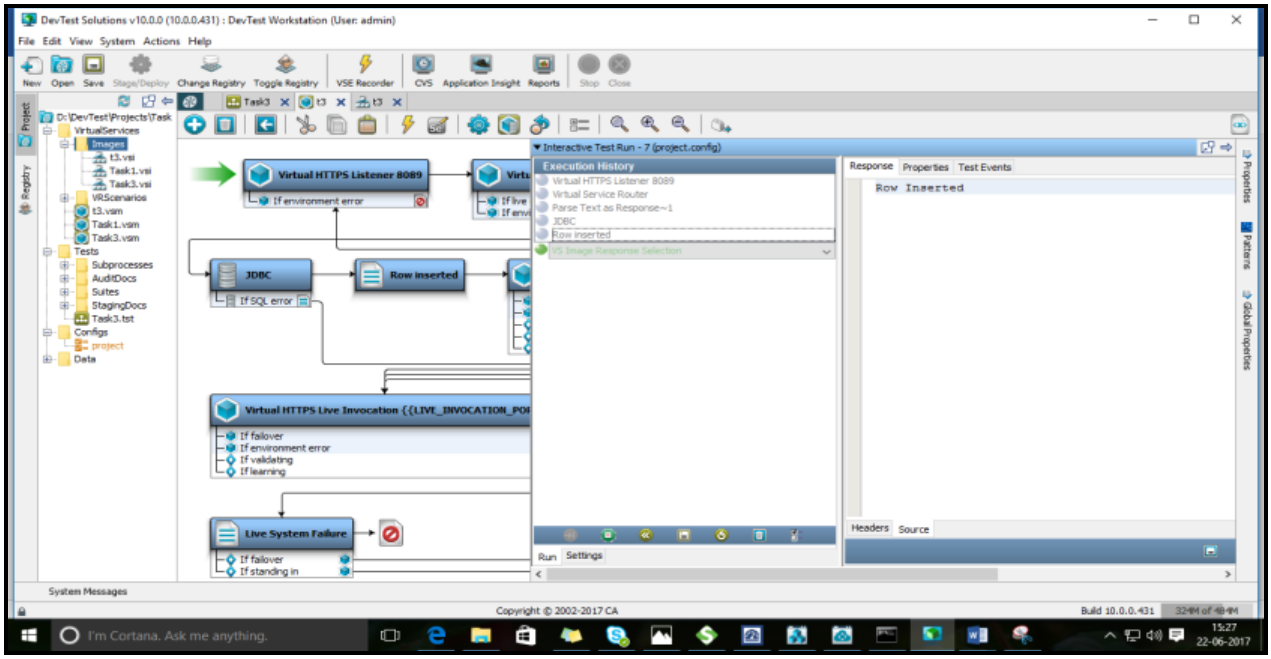
13) Add another **Output Log Message** step to display "**Record Exists**". If user tries to insert the record which is already present in db table.



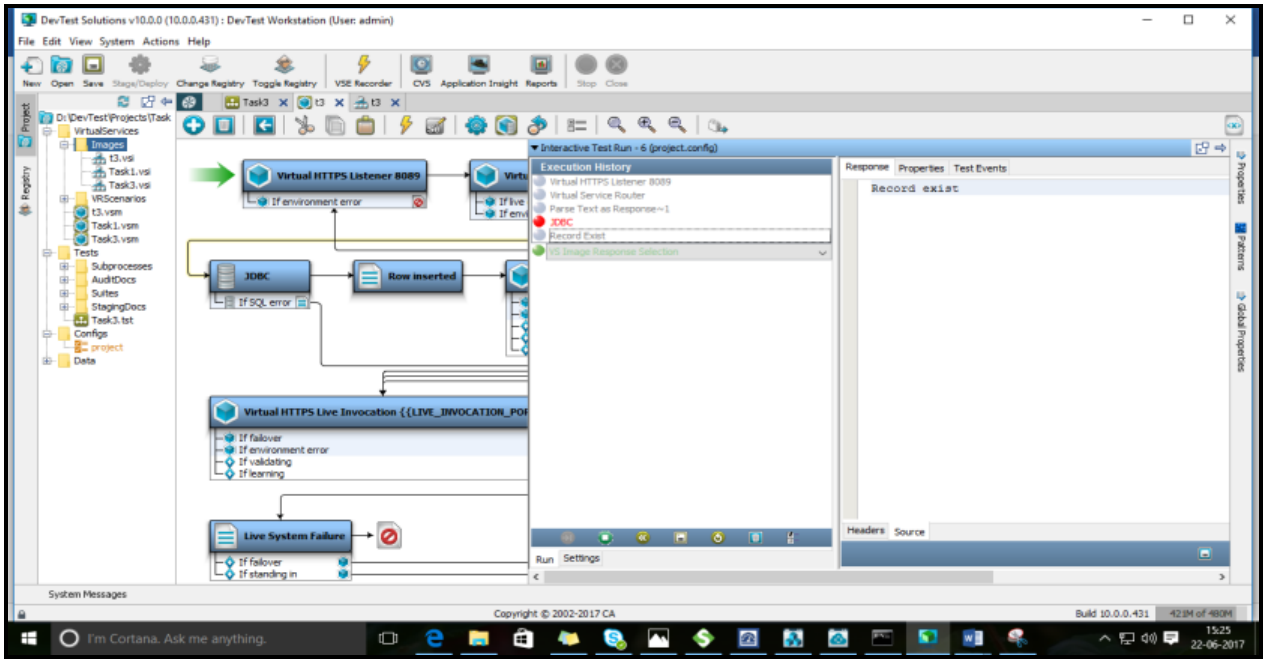
14) Rename the second output log message, right-click on **If SQL error** under **JDBC step** and select **goto renamed output log message step** (Ex: Duplicate Record Exists)



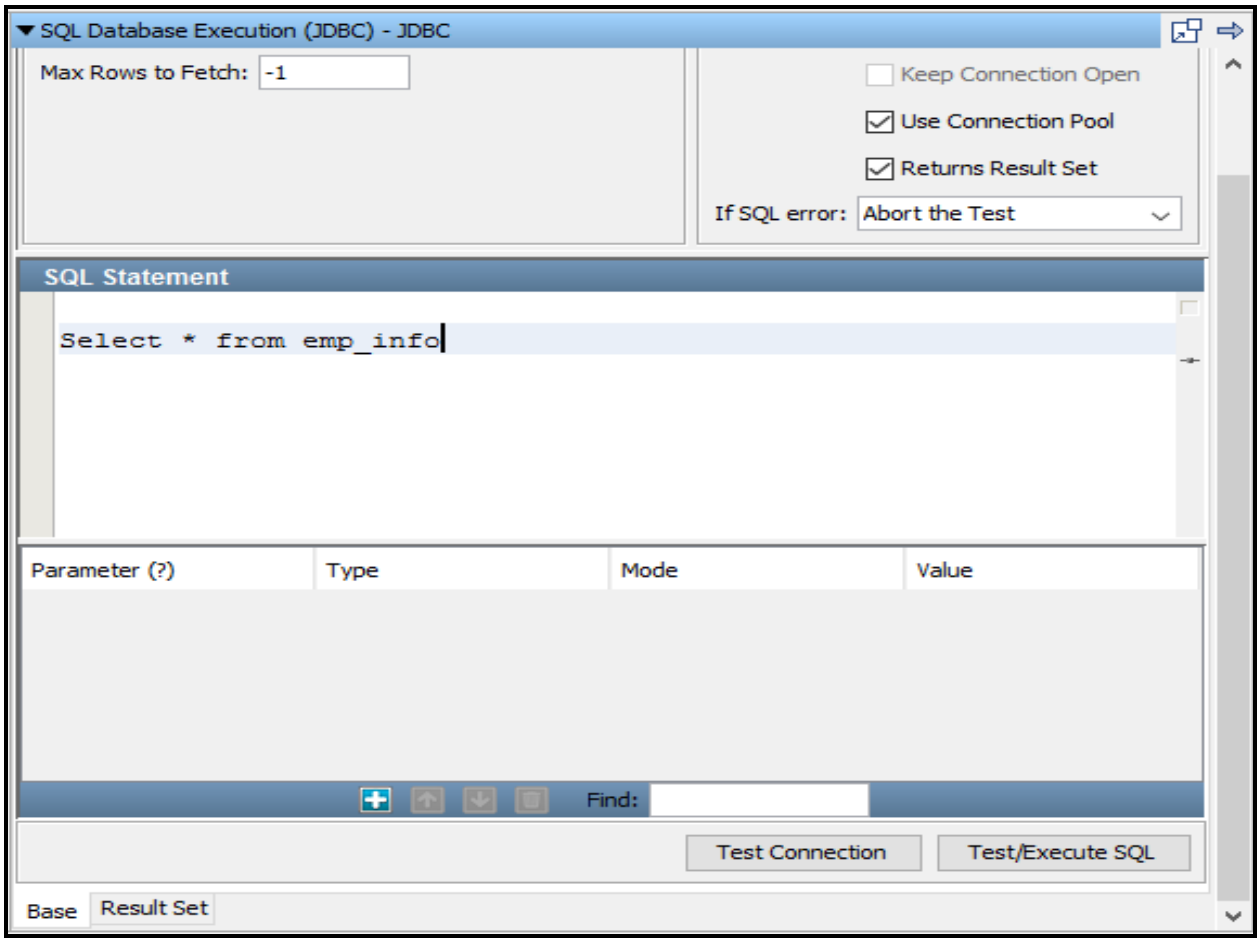
15) Save and run ITR. The Record is inserted in DB table, then a message “Row inserted” displayed.



16) If the Record is already present in DB table, then a message “Record exist” displayed.



17) To check whether DB is updated with records, Go to JDBC test step > write select query and Click on Test /Execute SQL



Note: Check Returns Result Set

Result Set

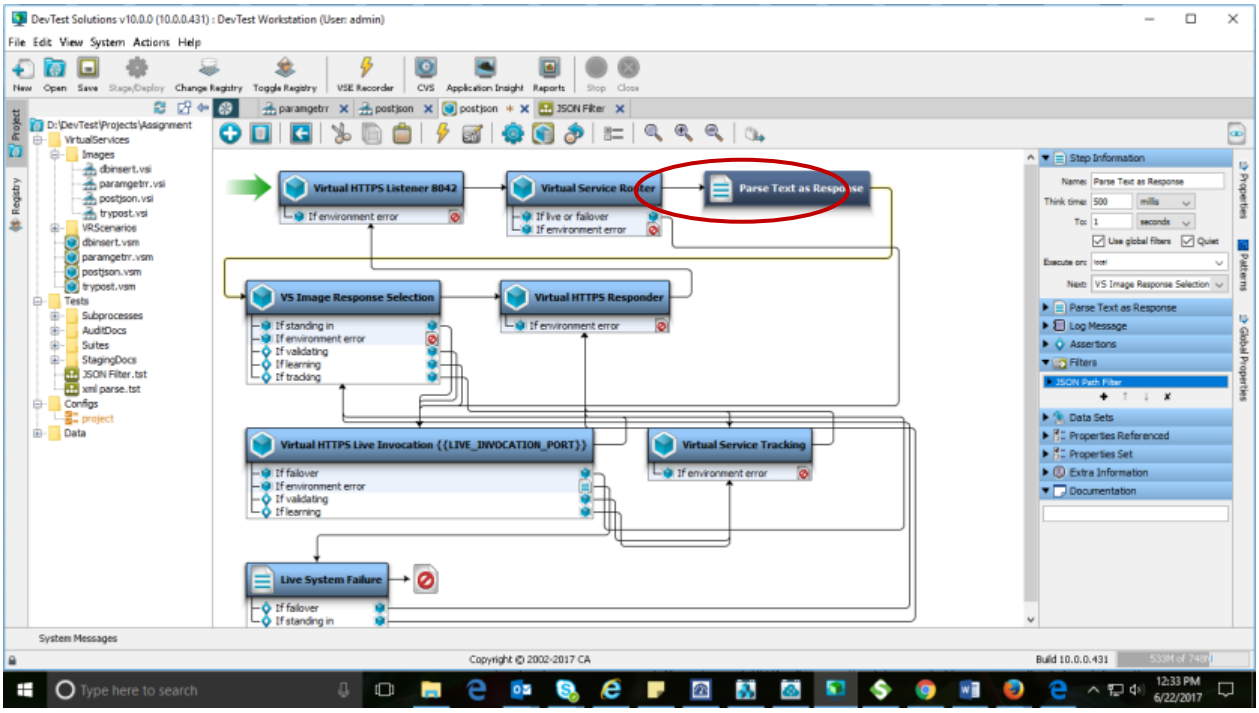
EMAILADDRESS	FIRSTNAME	LASTNAME	PASSWORD	USERNAME
abc@test.com	nbvc	tyrf	erty	erft_poi
hgty@test.com	tyui	iujk	hgfr	dfer_poi

Request and Response Files

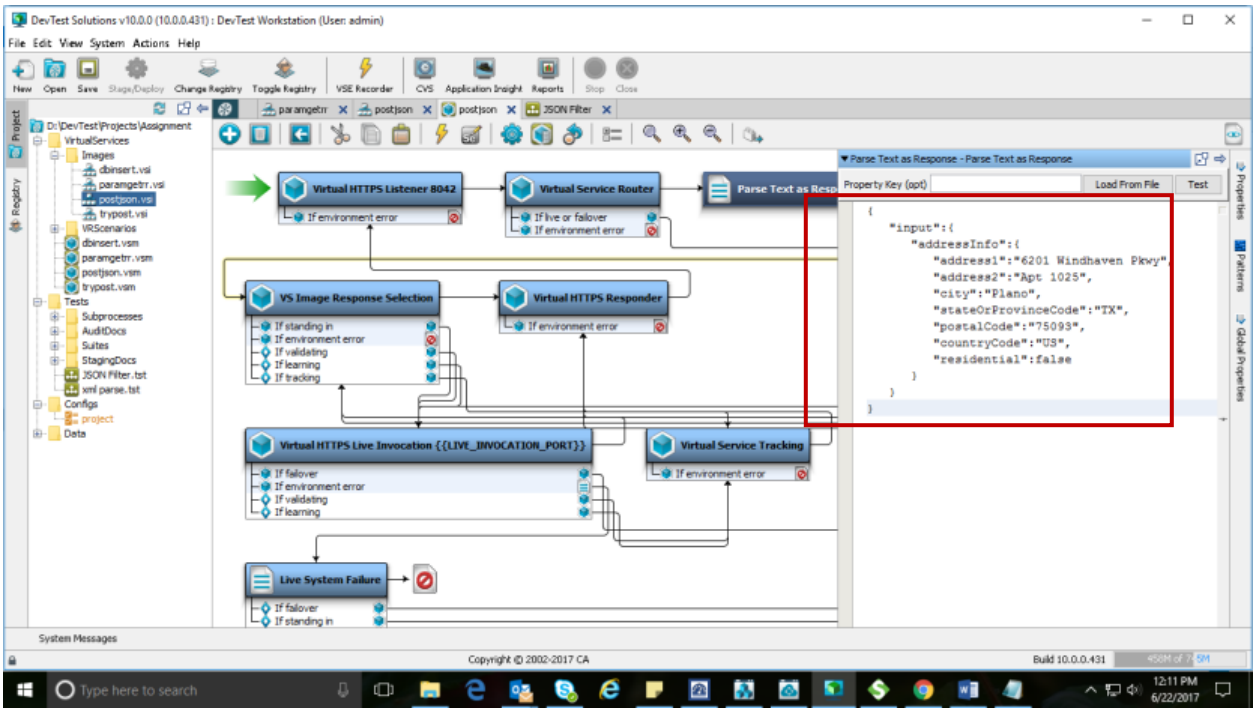
23.3 Capturing Input Parameter Values & Storing it in Different Variables

This scenario deals with capturing Input Parameter Values & Storing it in Different Variables Using REST Service POST Method

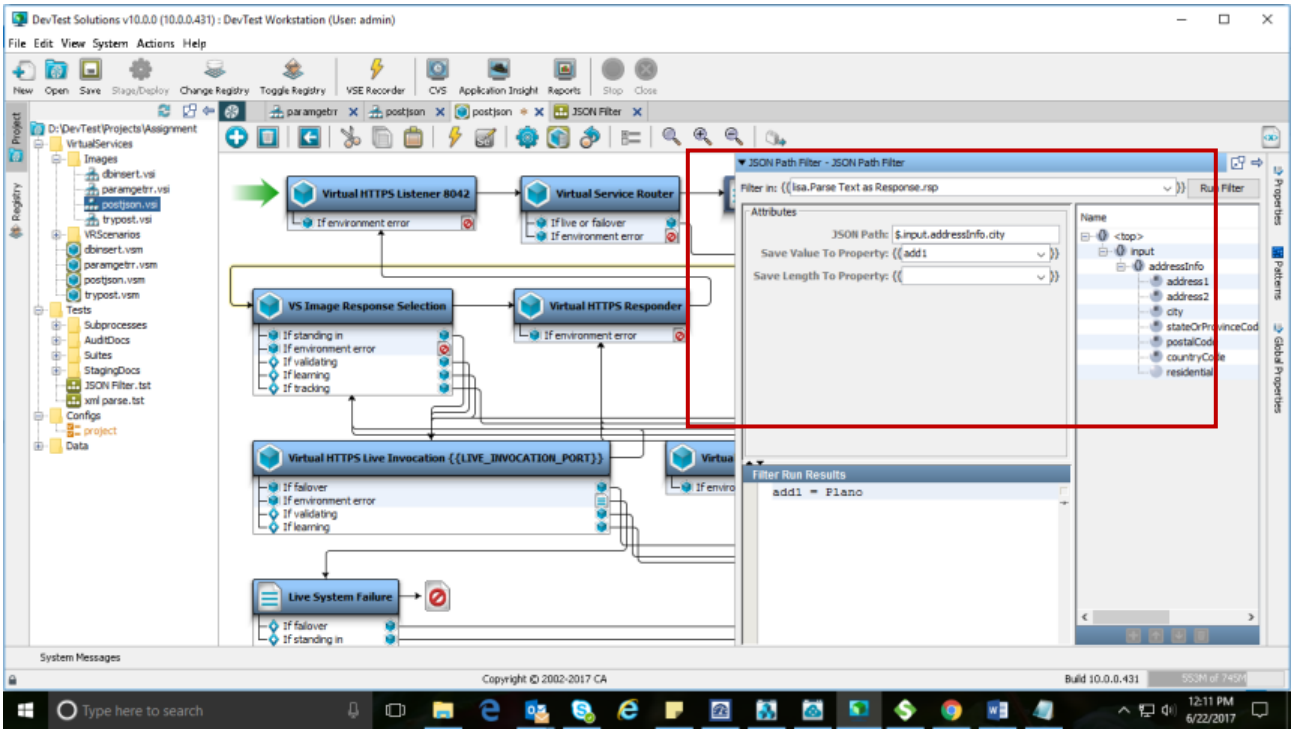
1) Create VSI & VSM by request-response pair
2) Open VSM and add Parse Text as a Response step in model editor



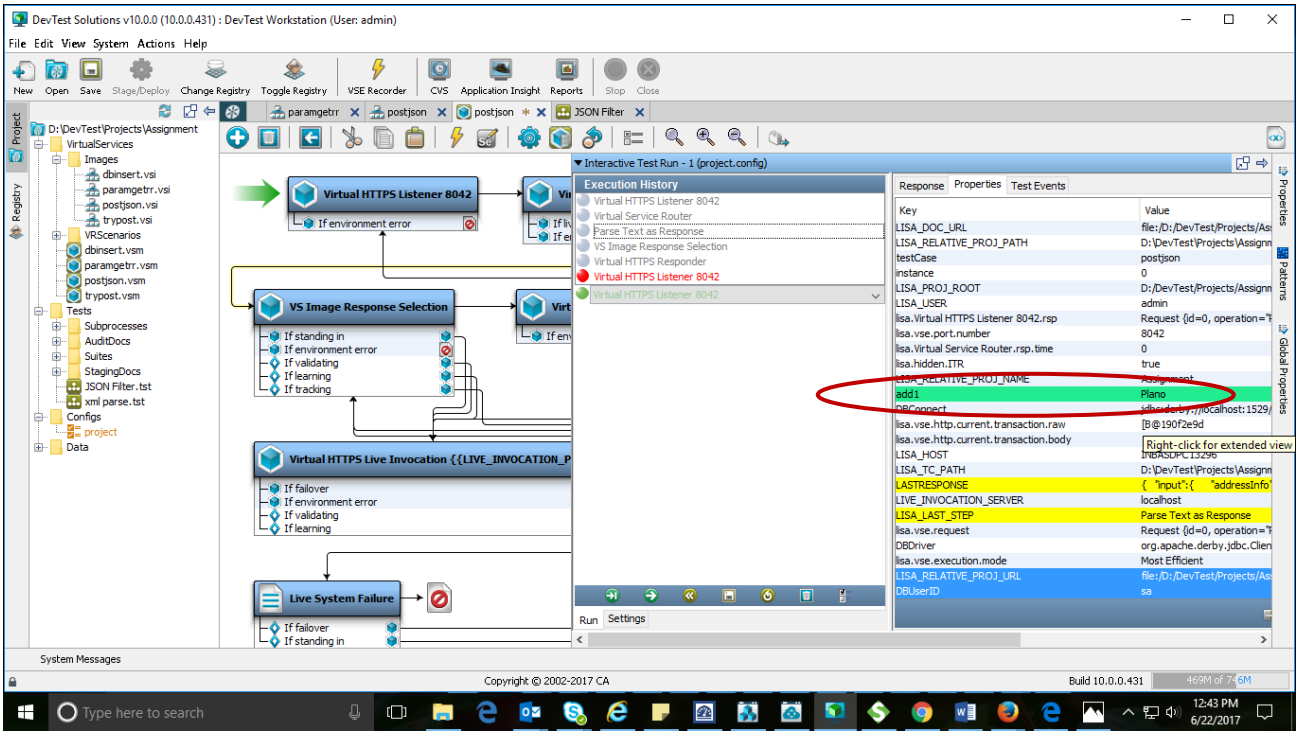
3) Click on Parse Text as a Response and copy request input into it



4) Add JSON Path Filter to the step and fill the following fields and run filter



5) Save and run the ITR



Input Parameter Values are stored in different variables Using REST Service POST Method