

Integrated test client

Objectives	2
Requirements	2
Skills acquired.....	2
Introduction	2
Part 1: Importing the projects	3
Part 2: Component testing.....	9
Part 3: Create test case and save test	17
Part 4: Testing the entire application	25
Part 5: Running a test suite	43
Part 6: Using the component test explorer	46

Objectives

WebSphere Integration Developer V6 brought various new features that enhance your ability to test modules. This lab will cover the following features updated for WebSphere Integration Developer V7:

- Sharing a data pool
- Human task emulation
- Fine grained trace
- Adding wait for time to tests
- Component test explorer

Requirements

- WebSphere Integration Developer V7 installed on your machine
- Project Interchange: LoanAppPI.zip

Skills acquired

- An understanding of WebSphere Integration Developer for testing modules
 - The ability to perform unit testing of a module or set of modules
 - The ability to perform component testing
 - The ability to add test cases, suites and emulations
 - An understanding of the Component Test Explorer

Introduction

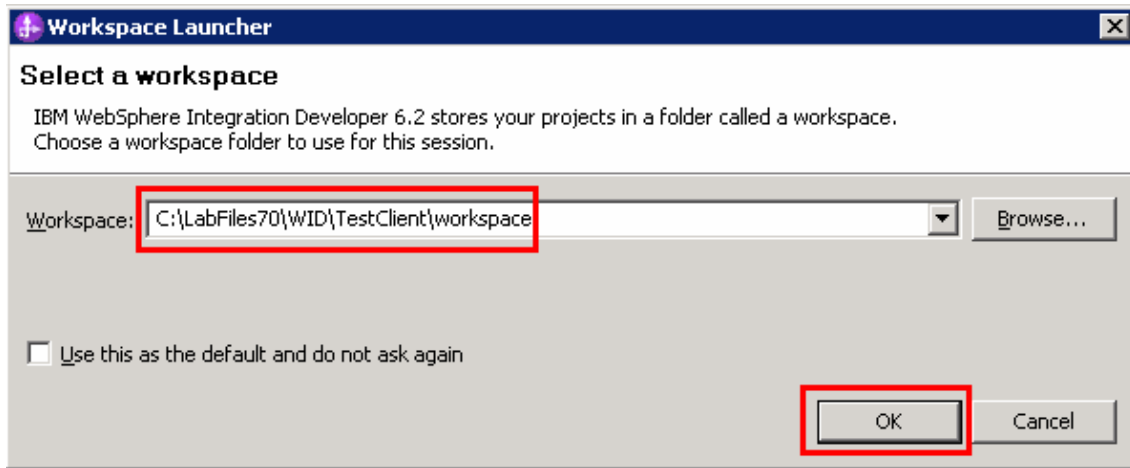
Using WebSphere® Integration Developer, you can test your modules by performing either unit testing or component testing. In *unit testing*, you choose the components and interfaces that contain the operations that you want to test, and then you test the operations one at a time in the integration test client. This is the form of testing that has been used with the integration test client before the version 7 release of WebSphere Integration Developer. In *component testing*, you use the new test suite editor and associated wizards to create and define test cases that consist of one or more operations. This enables you to run a test through several applications in the integration test client.

You will import a slightly modified version of the loan application. An extra routing mediation module is added which invoke the loan application module to make the lab more interesting.

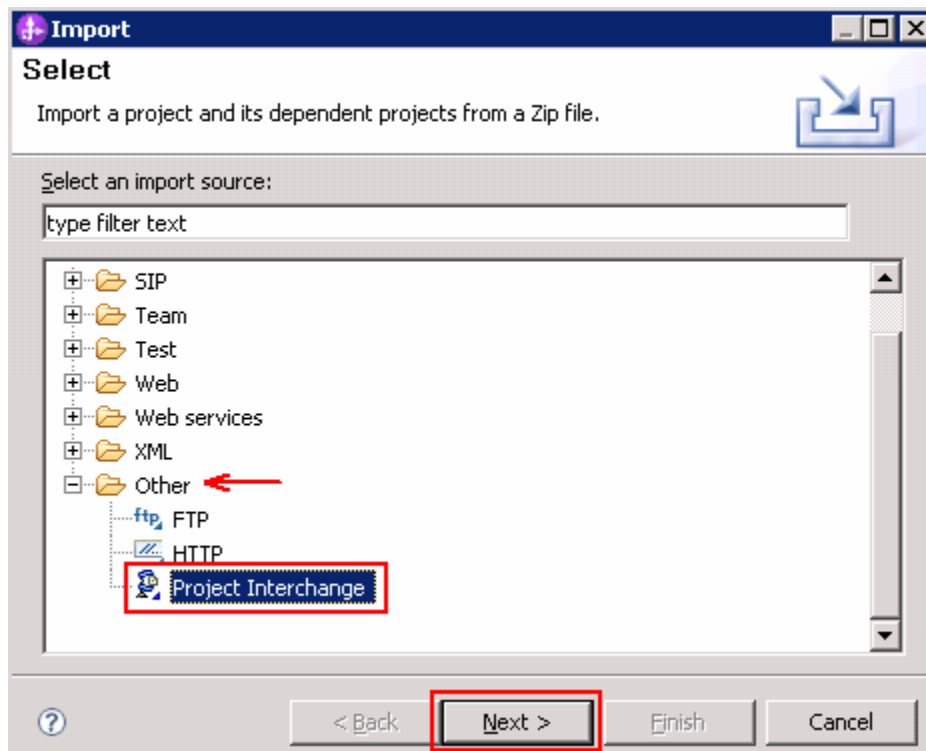
Part 1: Importing the projects

In this part of the lab, you will import the Loan Application project interchange to a new workspace. Eventually, you will examine all the projects and modules.

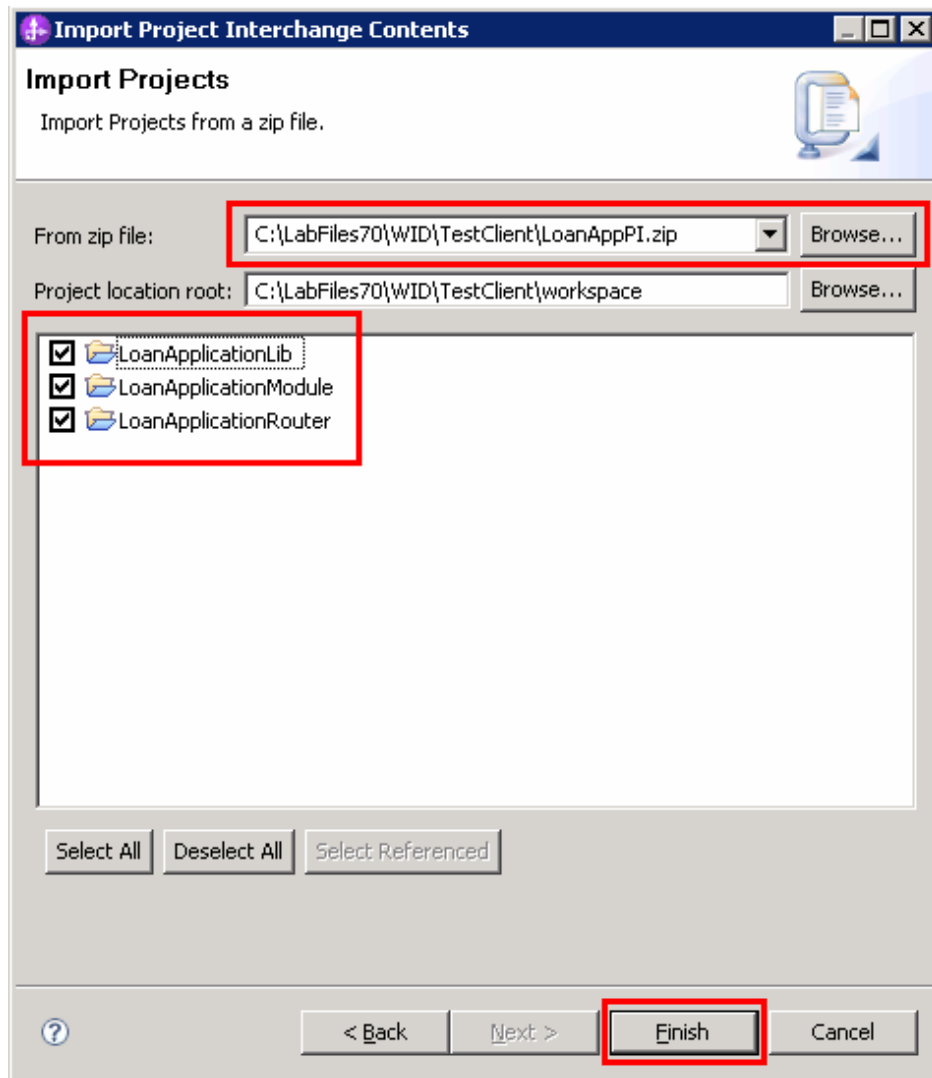
- ___ 1. Import the Loan Application project interchange to a new workspace
 - ___ a. Launch the WebSphere Integration Developer to a new workspace, for example
C:\Labfiles70\WID\TestClient\workspace



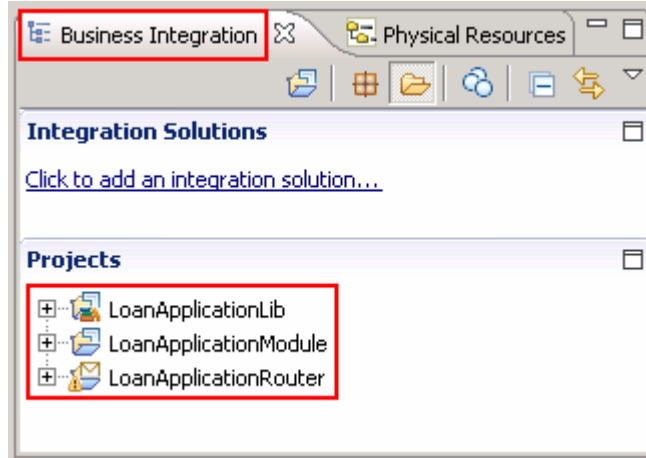
- ___ b. From the main menu, select **File** → **Import**
- ___ c. From the **Import** dialog, select **Other** → **Project Interchange**



- ___ d. Click **Next**
- ___ e. In the next **Import Projects** dialog, click **Browse** to select the Loan Application project (**LoanAppPI.zip**) interchange



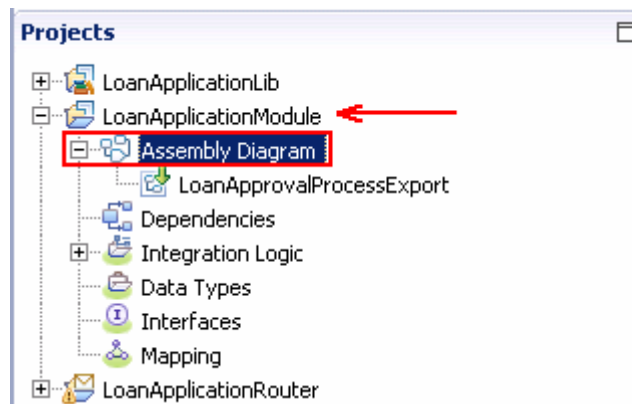
- ___ f. Click **Select All** to select all the modules listed from the project interchange
- ___ g. Click **Finish**.
- ___ h. The imported projects should be listed in the Business Integration shown below:



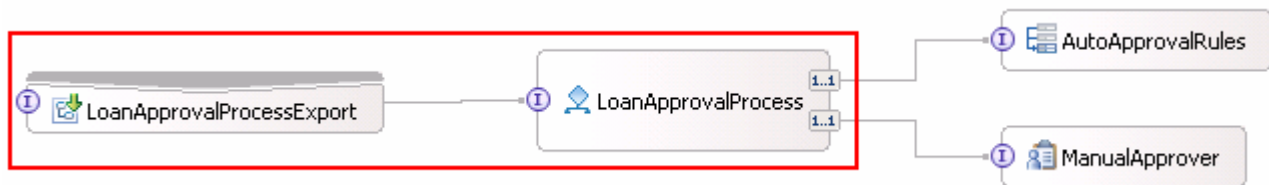
___ 2. Examine the projects

- ___ a. Take a look at the assembly diagram for the **LoanApplicationModule**. This is a service which approves a request for a loan. It is a slightly modified version of the loan application sample that comes with WebSphere Integration Developer.

In the business integration view, expand **LoanApplicationModule** and double click **Assembly Diagram** to open the assembly editor:

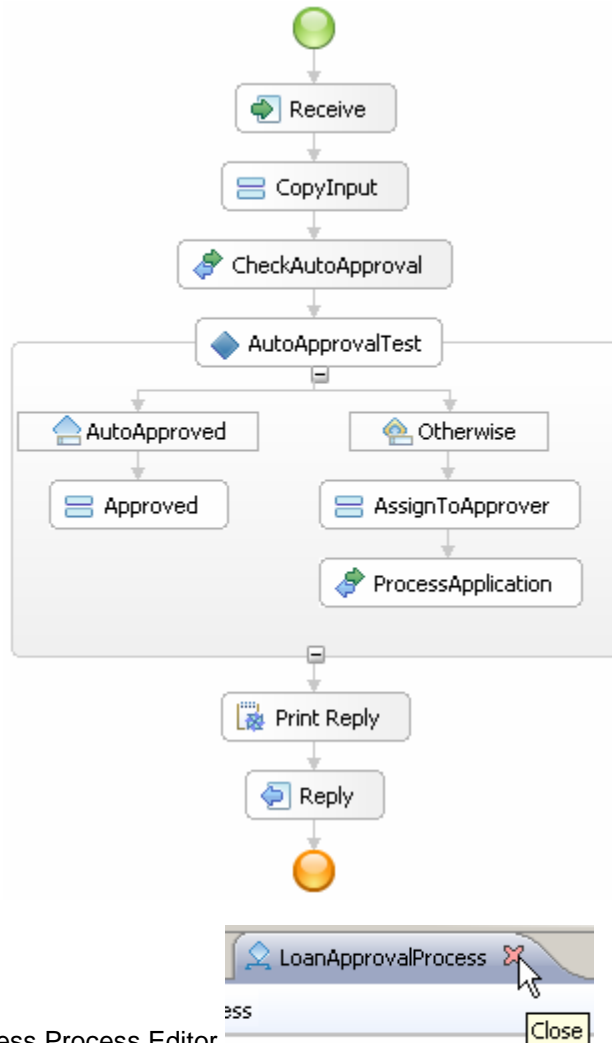


- ___ b. Notice there is an SCA Export connected to a BPEL process called **LoanApprovalProcesses**:



- ___ c. In the assembly diagram, double-click the **LoanApprovalProcess** icon to open the process editor. This is a simple process which invokes a business rule which decides whether the loan is approved. If the business rule decides the loan is not approved automatically, the process then invokes a human task for manual approval.

Note: Alternatively, you can open the **LoanApprovalProcess** in the process editor from the business integration view's project explorer from **LoanApplicationModule** → **Integration Logic** → **Processes** → **LoanApprovalProcess**



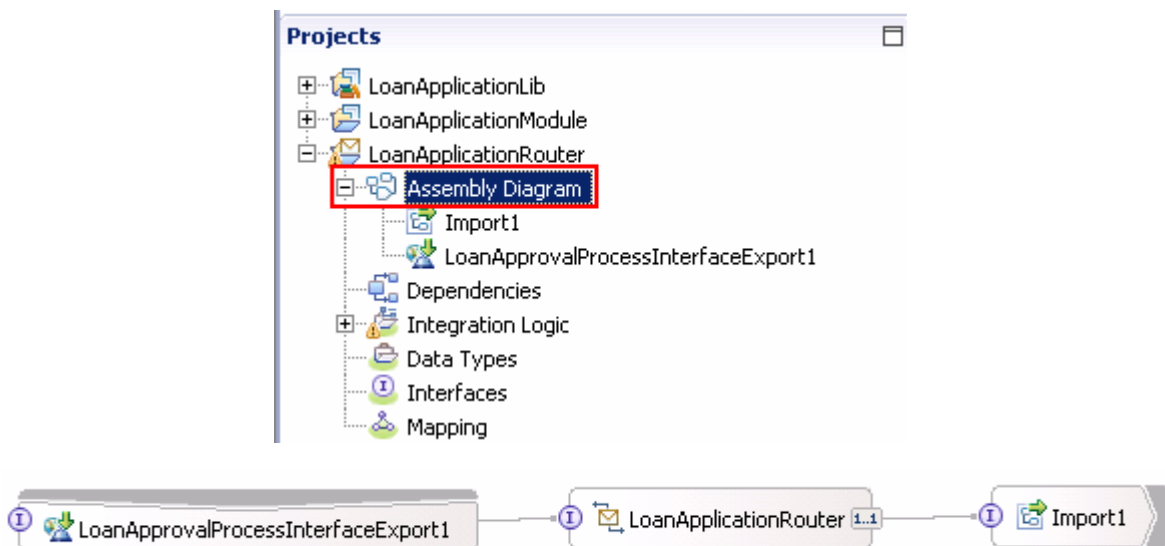
___ d. Close the Business Process Editor

___ e. By now you are probably questioning, what are the conditions for auto-approval? In the **Business Integration** view, expand **LoanApplicationModule** → **Integration Logic** → **Rule Logic** and double click the **AutoApprovalRequest** to open it in the rule set editor. Under the **Rules** section, note that the loans are automatically approved if the request is for less than 50,000:

▼Rules	
Name	Rule1
Presentation	
Action	AutoApprovalResponse = false
Name	Rule2
Presentation	
If	any of the following is true <ul style="list-style-type: none"> ■ AutoApprovalInput.LoanAmount <=50000
Then	AutoApprovalResponse = true

Note: If the request is more than 50,000 the loan is not automatically approved. A human task is used to approve the loan if over 50,000.

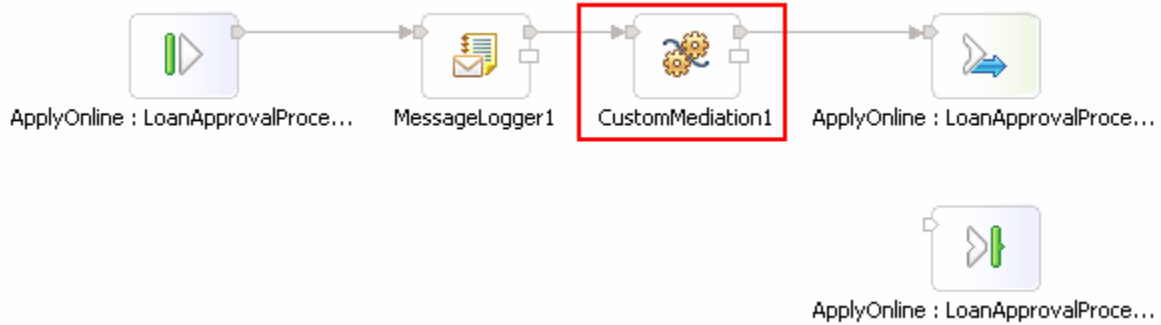
- ___ f. .Close the rule set editor.
- ___ g. Close the assembly editor.
- ___ h. View the **LoanApplicationRouter** mediation module. In the business integration view, expand **LoanApplicationRouter** and double click double click **Assembly Diagram** to open it in the assembly editor:



Note: This module's purpose is for demonstration. Later on in the lab you will learn how to attach two modules to the test client in order to test them both at the same time.

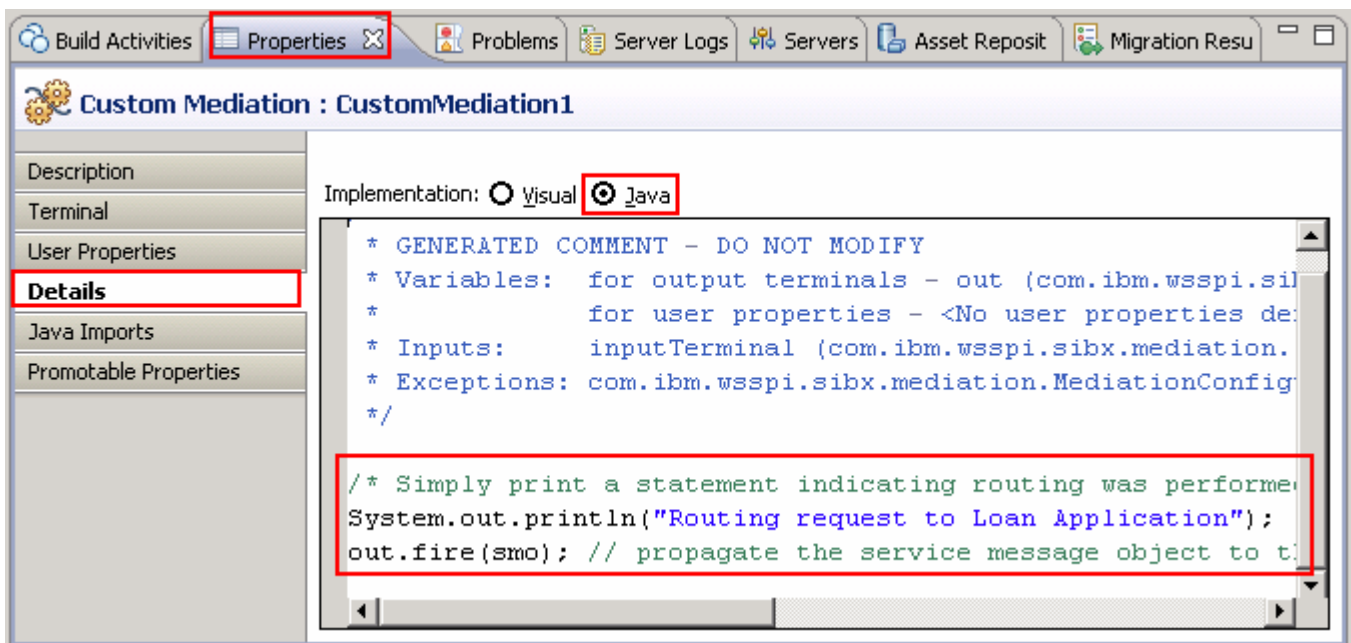
Note: The import **Import1** is an SCA import which invokes the export **LoanApprovalProcessInterfaceExport1** in the **LoanApplicationModule**.

- ___ i. In the assembly diagram, double-click the **LoanApplicationRouter** component to open it in the Mediation Flow Component editor



__ j. In the mediation flow editor, right click **CustomMediation1** and select **Show in Properties**. In the CustomMediation1 properties view, select the **Details** tab to view the **Java** implementation as shown below:

Note: The mediation flow logs the input message and invokes a custom mediation to print a statement.

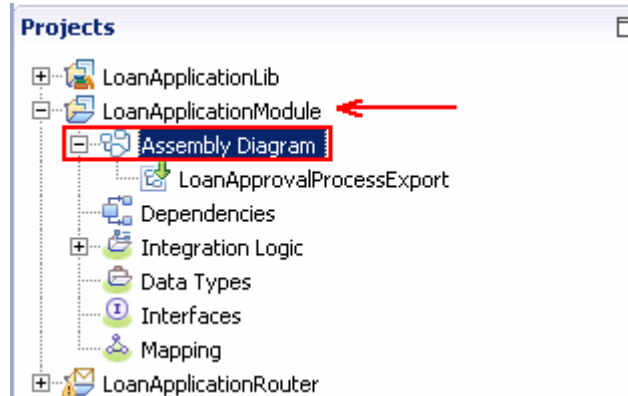


__ k. Close the mediation flow editor

Part 2: Component testing

In this part of the lab you test individual components. For example, you just finished implementing the **AutoApprovalRules** component in the **LoanApplicationModule**. Before continuing you may want to test that specific component to ensure the business logic works properly.

- ___ 1. Testing the **AutoApprovalRules** component
 - ___ a. In the business integration view, expand **LoanApplicationModule** and double click **Assembly Diagram** to open the assembly editor:



- ___ b. In the **LoanApplicationModule** assembly diagram, right-click the **AutoApprovalRules** component and select **Test Component in Isolation**. This opens the Test Client:

Events

General Properties

Detailed Properties

Configuration: Default Module Test

Module: LoanApplicationModule

Component: AutoApprovalRules

Interface: AutoApprovalRulesInterface

Operation: AutoApprovalRequest

Initial request parameters

Name	Type	Value
AutoApprovalInput	LoanRequestBO	✓
ApplicantInfo	PersonalDataBO	✓
Name	string	✓
EmailAddress	string	✓
TaxPayerId	int	✓ 0
LoanAmount	int	✓ 0

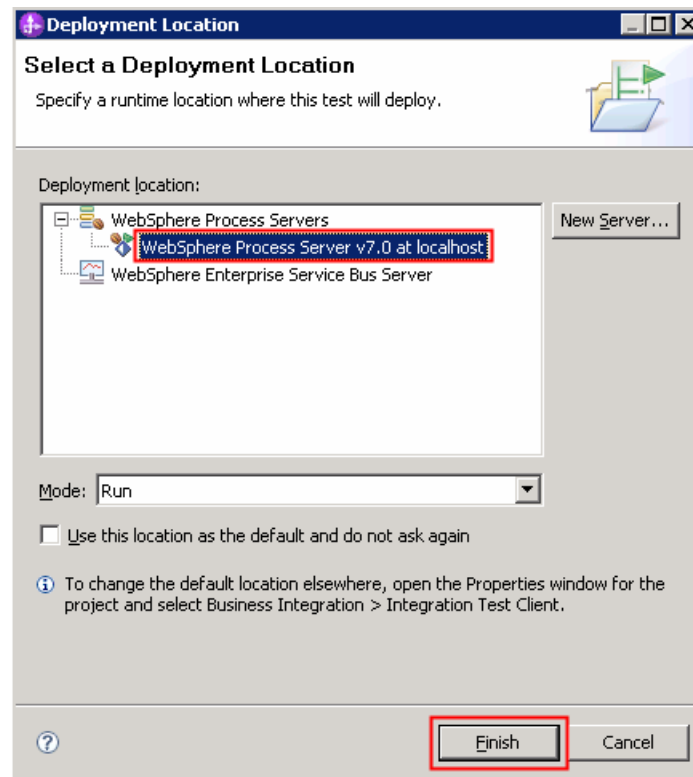
Note: The Test Client automatically selects the corresponding Module & Component. Since the component only has one interface and one operation, those are correctly populated for you.

___ c. Enter the following **Initial request parameters**:

Name : Joe
EmailAddress : Joe@ibm.com
TaxPayerId : 123456789
LoanAmount : 49999

___ d. Click the **Continue** button on the top left of the test client editor: (▶)

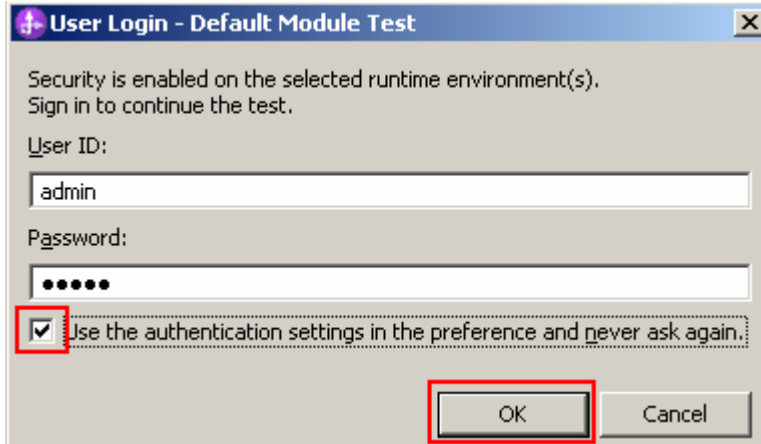
___ e. The **Deployment Location** dialog opens:



Note: Subsequent tests on this session of the test client may not prompt for **Deployment Location** selection. You can select “Use this as the default and do not ask again” check box if you know without a doubt all tests will use this runtime location.

___ f. Select **WebSphere Process Server v7** and then click **Finish**.

___ g. The **User Login** dialog opens:



- ___ h. Enter the **User ID** and **Password** for the WebSphere Process Server test server. To make subsequent testing faster, check the “Use the authentication settings in the preference and never ask again.” check box. Do not check if authentications need change with different tests. Then click **OK**.

Note: The WebSphere Process Server test server is automatically started if it is not started at this time and then publishes the module.

- ___ i. Examine the test results. The business rule should return **true** since the loan requested was for less than 50,000 as shown in the **Return parameters** section below:

Events

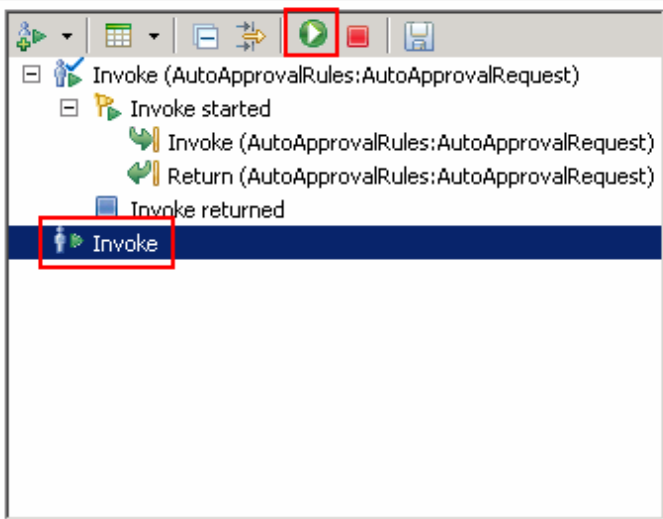
Module: [LoanApplicationModule](#)
 Component: [AutoApprovalRules](#)
 Interface: [AutoApprovalRulesInterface](#)
 Operation: [AutoApprovalRequest](#)

Return parameters:

Name	Type	Value
AutoApprovalResponse	boolean	✓ true

- ___ 2. Test for amounts that are greater than 50,000
 - ___ a. Click the **Invoke** button at the top left of the test client window: ()
 - ___ b. A new test entry appears with the input automatically populated for you. This is because the test client is smart and sees that you already entered data for this request. All you need to do is change the **LoanAmount** to 50,001:

Events



Component:

Interface:

Operation:

Initial request parameters ←

Name	Type	Value
AutoApprovalInput	LoanRequestBO	✓
ApplicantInfo	PersonalDataBO	✓
Name	string	✓ Joe
EmailAddress	string	✓ joe@ibm.com
TaxPayerId	int	✓ 1234567
LoanAmount	int	✓ 50001

- c. To run the test, click the **Continue** button on the top left of the test client editor. Notice that the business rule returns a value of **false** since the LoanAmount was for more than 50,000

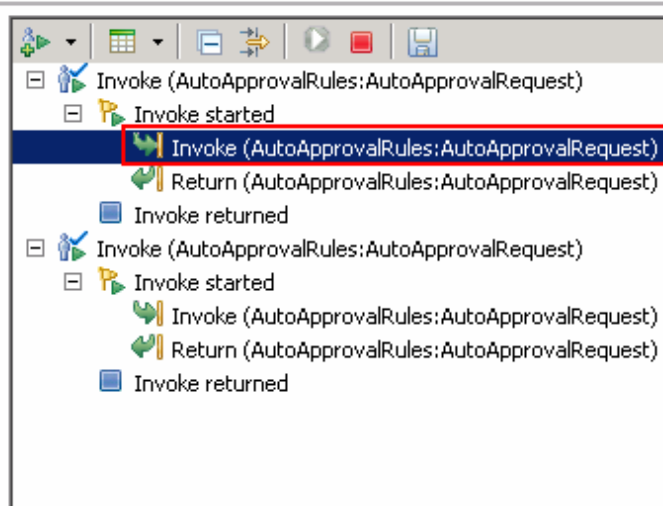
Return parameters:

Name	Type	Value
AutoApprovalResponse	boolean	✓ false

- 3. Export your test data in XML format. Have you ever wanted to see or save your inputted business object in XML format? Now you can export it directly from the test client editor.

- a. Expand the first **Invoke** → **Invoke started** and select **Invoke** in the test client editor.

Events



Configuration: [Default Module Test](#)

Module: [LoanApplicationModule](#)

Component: [AutoApprovalRules](#)

Interface: [AutoApprovalRulesInterface](#)

Operation: [AutoApprovalRequest](#)

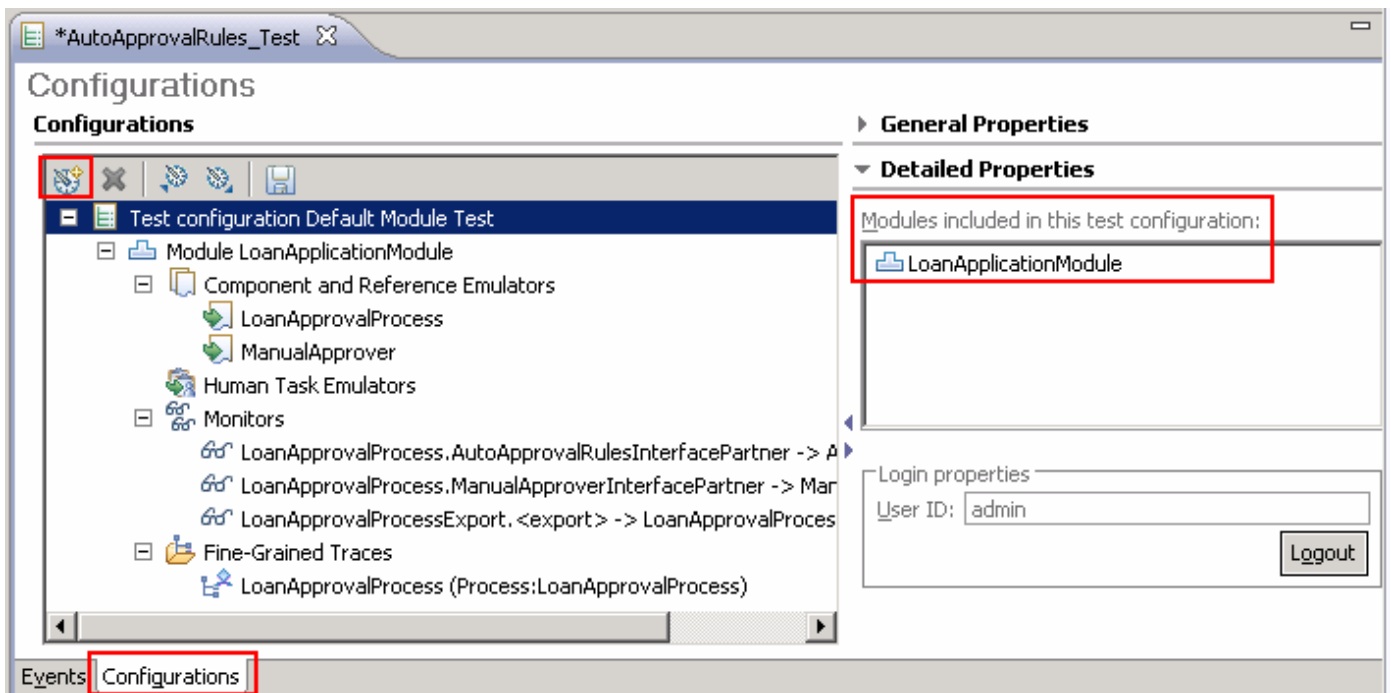
Invocation parameters

Name	Type	Value
AutoApprovalInput	LoanRequestBO	✓
ApplicantInfo	PersonalDataBO	✓
Name	string	✓ Joe
EmailAddress	string	✓ joe@ibm.com
TaxPayerId	int	✓ 123456789
LoanAmount	int	✓ 49999

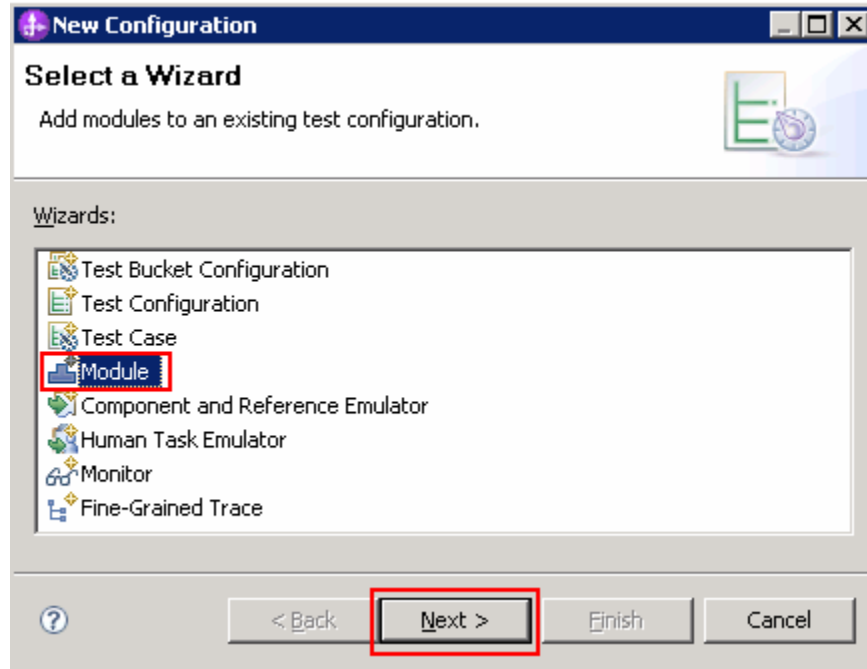
- ___ b. In the **Input Parameters** table, right-click **AutoApprovalInput** and select **Export to File...**
- ___ c. Save the file to the disk (Ex: C:\Labfiles62\WID\TestClient\workspace) with a meaningful name such as **AutoApprovalInput_TestData1.xml**
- ___ d. Now open the **AutoApprovalInput_TestData1.xml** file in the editor of your choice. It should resemble the following XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:AutoApprovalInput xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns
  <ApplicantInfo>
    <Name>Joe</Name>
    <EmailAddress>joe@ibm.com</EmailAddress>
    <TaxPayerId>123456789</TaxPayerId>
  </ApplicantInfo>
  <LoanAmount>49999</LoanAmount>
</ns0:AutoApprovalInput>
```

- ___ 4. Now test the entire application. Since the business rule component tests ran successfully, there is a good chance the rest of the application will run successfully also. Skip testing all of the components individually and test the application as a whole.
 - ___ a. Select the **Configurations** tab at the bottom of the test client editor.

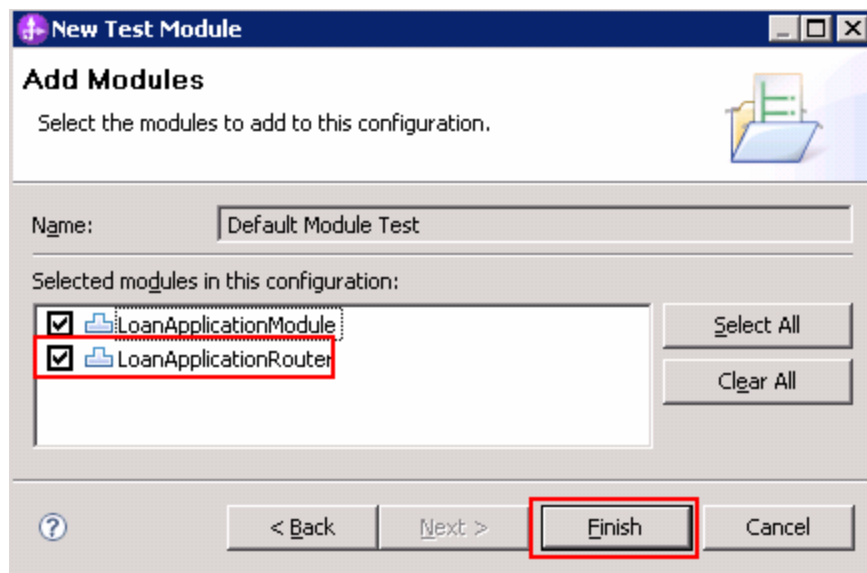


- ___ b. Click the **Add** button: ()
- ___ c. From the **New Configuration dialog**, select the **Module** wizard:




- ___ d. Click **Next** twice
- ___ e. Select the check box for **LoanApplicationRouter** module

Note: The **LoanApplicationModule** has been already selected during the previous individual module test and is running on the test server.

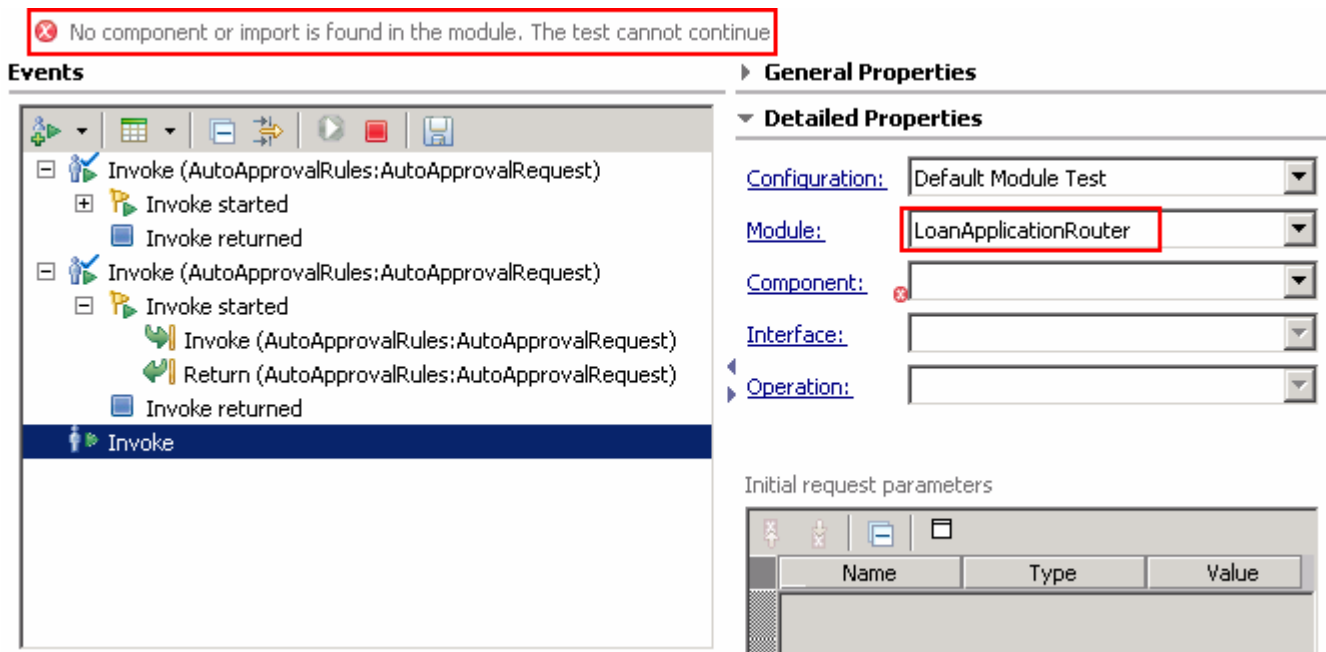


- ___ f. Click **Finish**. This action adds the **LoanApplicationRouter** mediation module to the test client. If you did not perform this step, the **LoanApplicationRouter** mediation module would have to be emulated. Instead, it will now invoke the module using the SCA import/export.
- ___ g. Now select the **Events** tab at the bottom of the test client editor

___ h. Click the **Invoke** button at the top left of the test client window: ()

___ i. Select **LoanApplicationRouter** module under Detailed Properties section

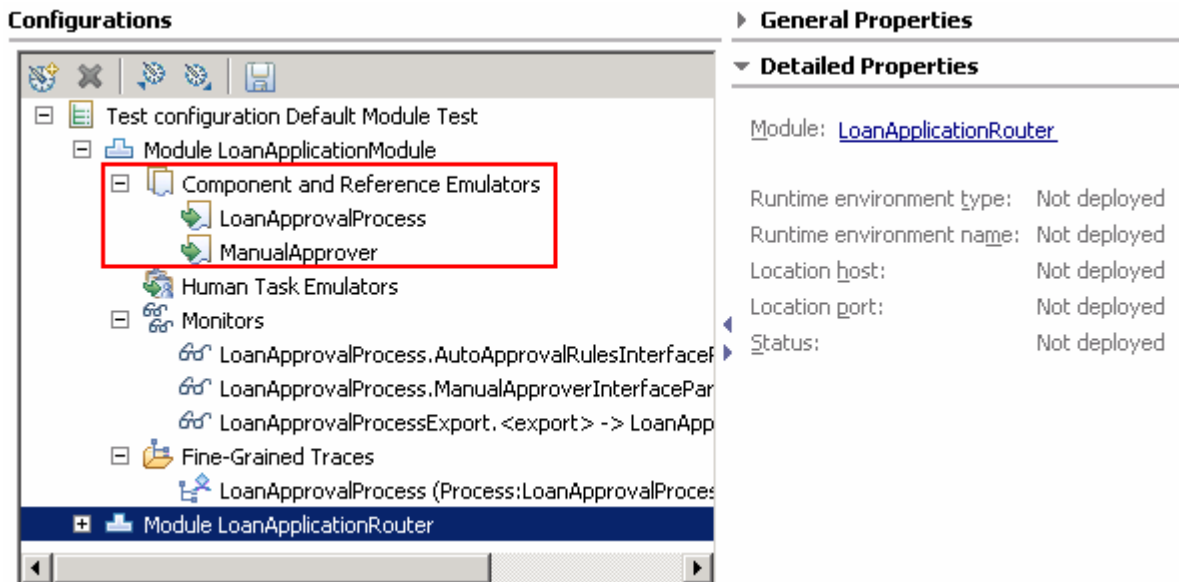
No component or import is found in the module. The test cannot continue



The screenshot shows the test client interface. At the top, a red-bordered error message states: "No component or import is found in the module. The test cannot continue". Below this, the 'Events' panel on the left shows a tree view of test events, including 'Invoke (AutoApprovalRules:AutoApprovalRequest)' and its sub-events like 'Invoke started', 'Invoke returned', and 'Return'. The 'General Properties' panel on the right shows the 'Module' dropdown set to 'LoanApplicationRouter', which is highlighted with a red box. Other dropdowns for 'Configuration', 'Component', 'Interface', and 'Operation' are also visible. At the bottom right, there is a table for 'Initial request parameters' with columns for Name, Type, and Value.

Notice that the test client complains that no component or an import is found and so the test cannot continue. Investigate the reason behind why the component is missing.

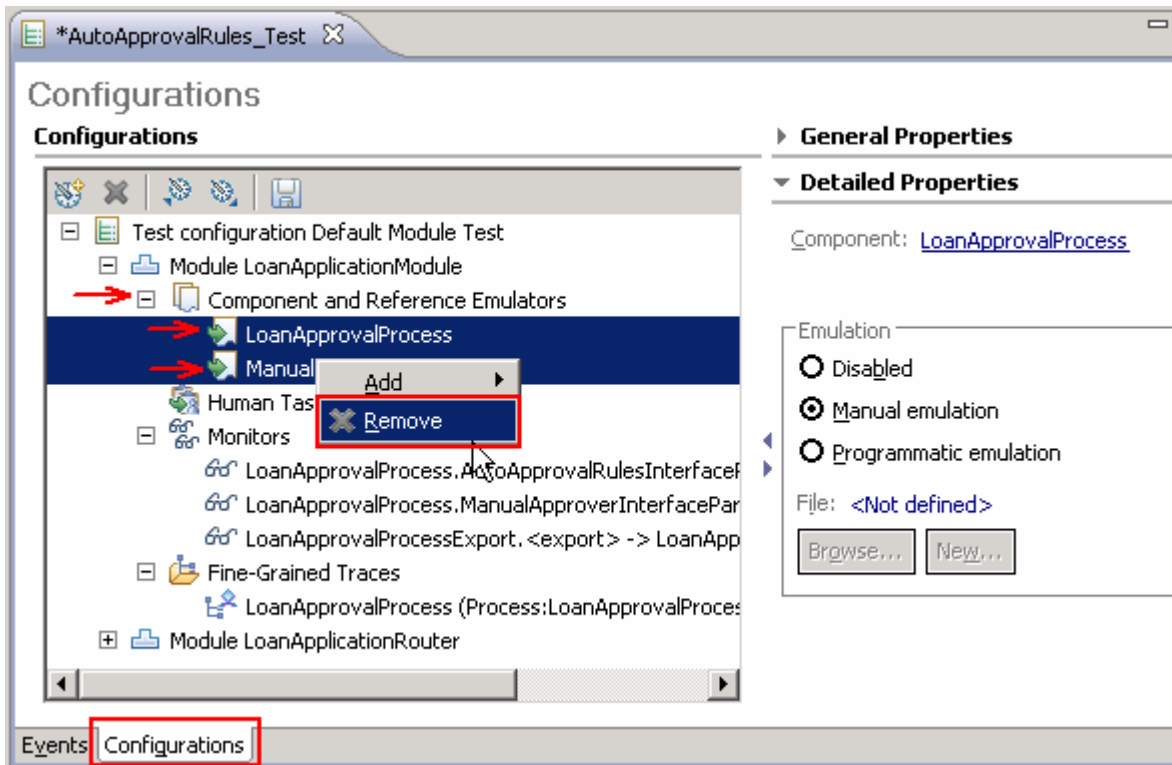
___ j. Click the **Configurations** tab at the bottom left of the test client editor. Look closely at the configuration for the **LoanApplicationModule**. Notice there are emulators for certain components.



The screenshot shows the 'Configurations' panel on the left, which displays a tree view of test configurations. The 'Component and Reference Emulators' folder is highlighted with a red box, showing sub-items like 'LoanApprovalProcess' and 'ManualApprover'. The 'Detailed Properties' panel on the right shows the 'Module' set to 'LoanApplicationRouter'. Below this, several properties are listed, all with a value of 'Not deployed': 'Runtime environment type', 'Runtime environment name', 'Location host', 'Location port', and 'Status'.

This is because you started the test client by choosing to test the **AutoApprovalRules** component *in isolation*. The test client adds emulators for all other components in the module when you select this option.

- ___ k. Remove the emulators. While the **configuration** tab of the test client is selected, hold the **Ctrl** button and select all the modules, right click and then select **Remove** as shown below:

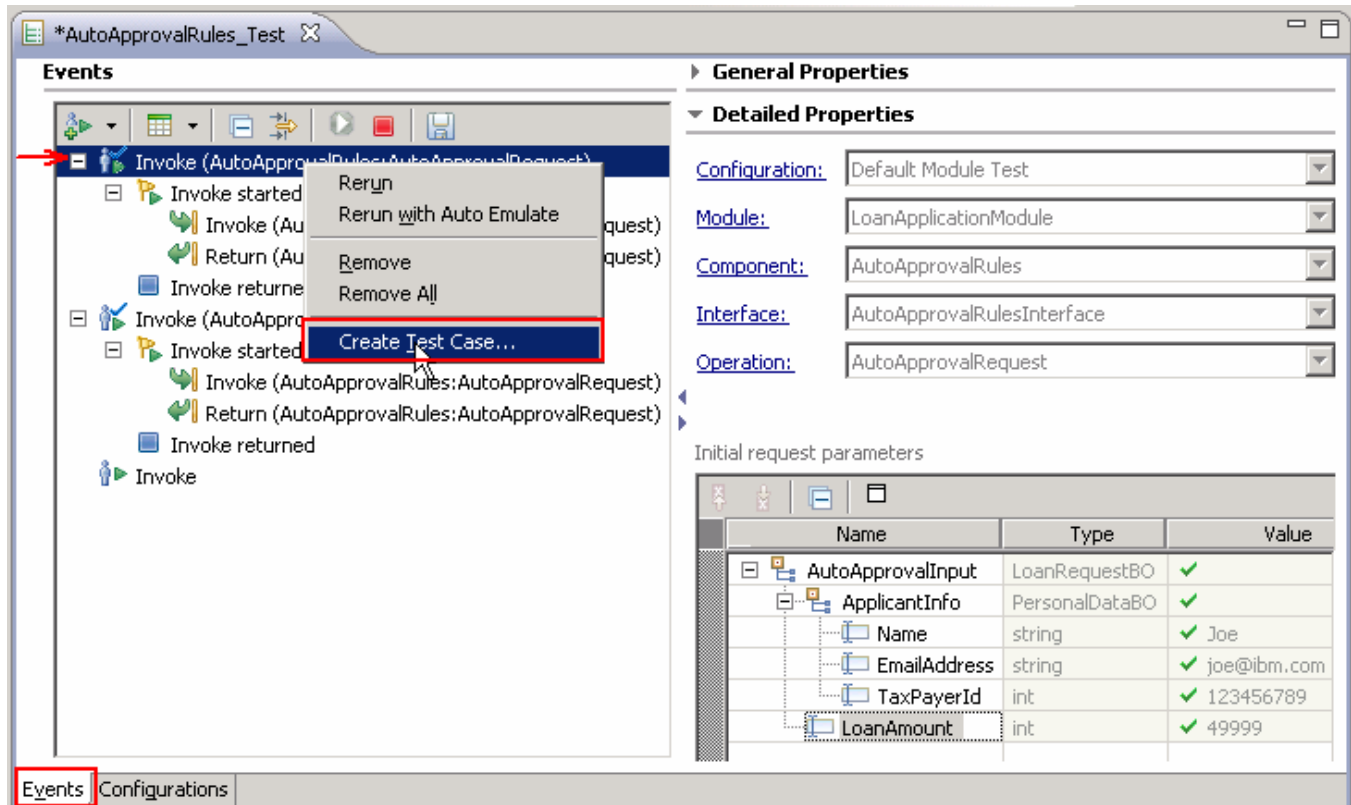


Note: To test all the components, you would have to go back to the Assembly Diagram and right click in the white space of the editor and select **Test Module** from the menu. You can also right click **AutoApprovalRules** component in the Assembly Diagram of the **LoanApplicationModule** and select **Test Component** instead of **Test Component in Isolation**.

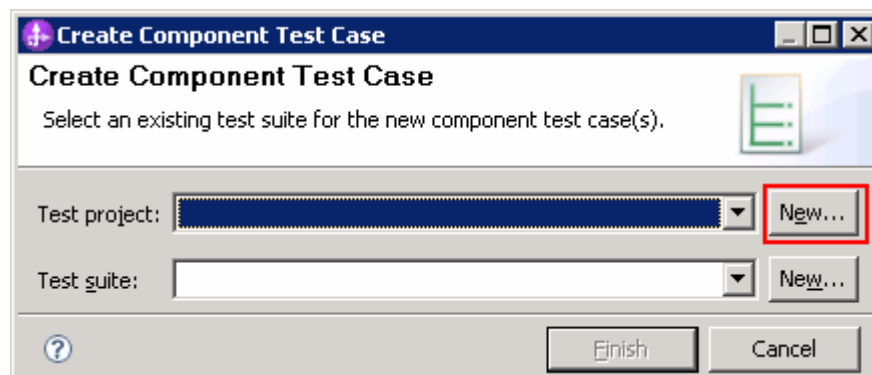
Part 3: Create test case and save test

In this part of the lab, you will save the work you have done to so you can rerun the test at a later time.

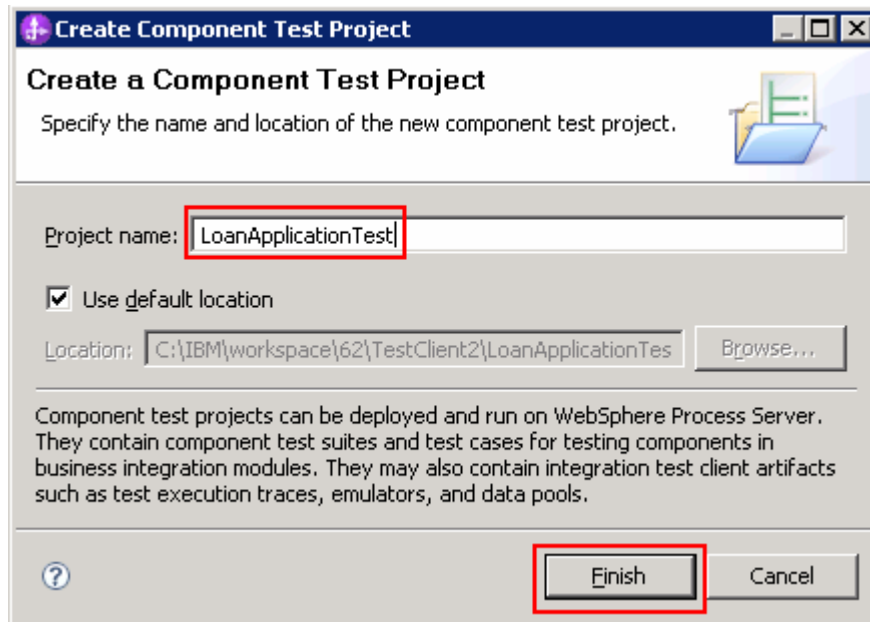
- ___ 1. Creating a test project, test suite and test case.
 - ___ a. Click the **Events** tab at the bottom of the test client editor.
 - ___ b. Right-click the first **Invoke** in the test client and select **Create Test Case** from the menu



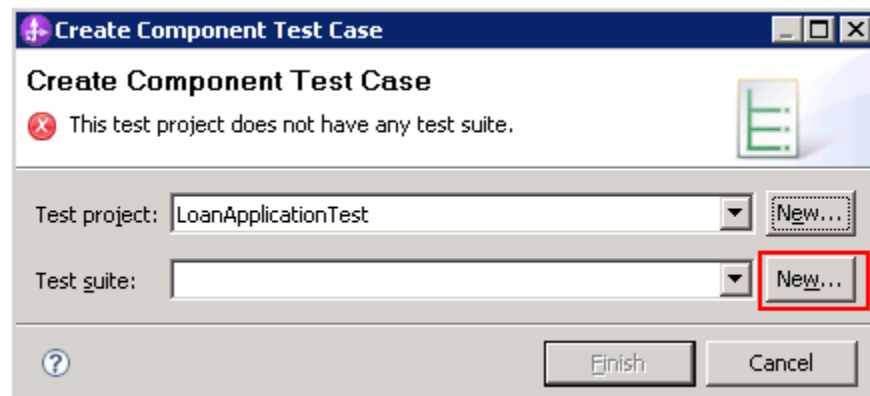
- ___ c. The **Create Component Test Case** dialog opens



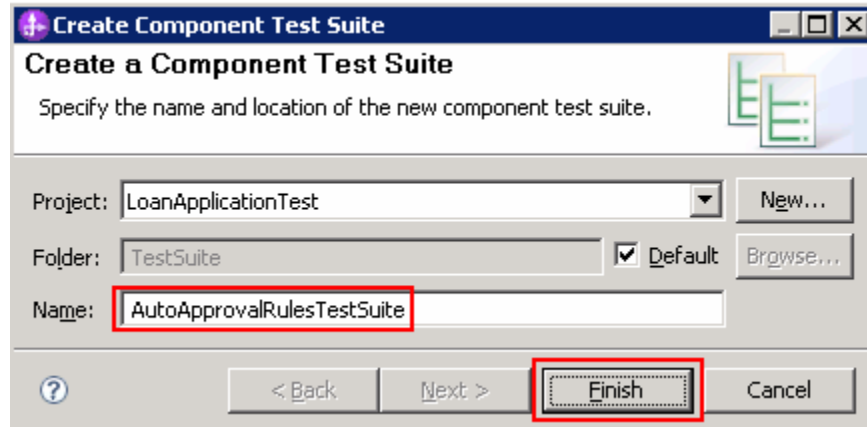
- 1) Click **New** for **Test Project** to create a new Component Test Project named, **LoanApplicationTest**



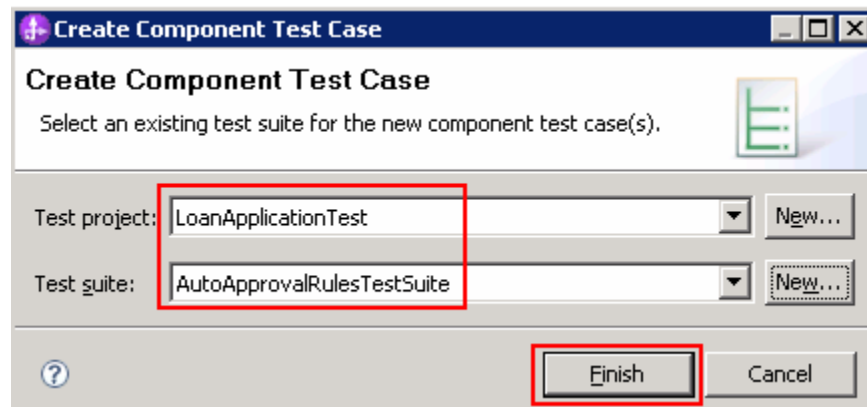
- 2) Click **Finish**



- 3) Now click **New** for **Test suite** to create a new Component Test Suite named, **AutoApprovalRulesTestSuite**

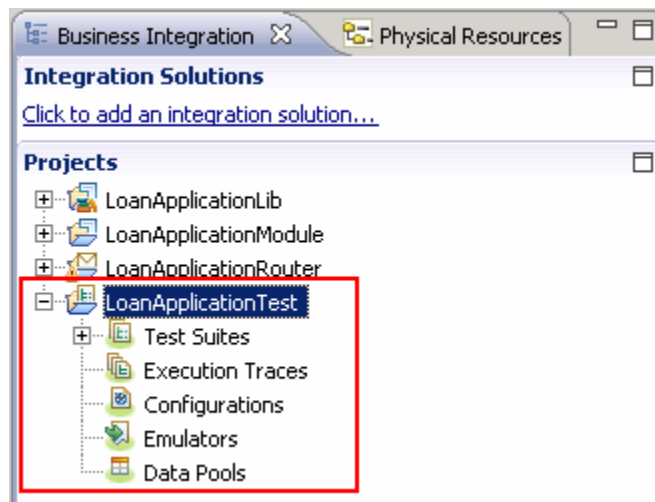


4) Click **Finish**

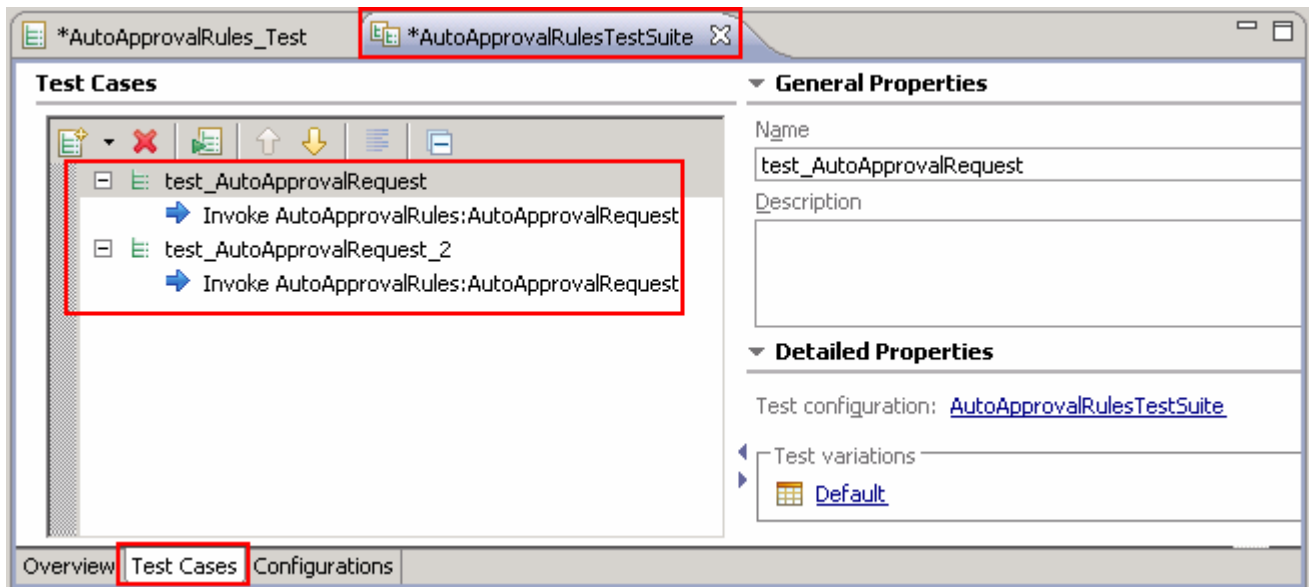


___ d. Click **Finish**

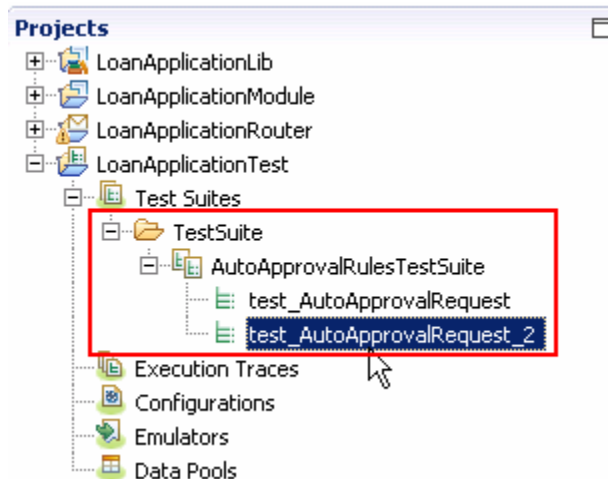
___ e. Notice the following new artifacts are created in the **Business Integration** view. In the Business Integration view, expand **LoanApplicationTest**



- ___ f. Similarly, right-click the second **Invoke** in the test client and select **Create Test Case** from the menu. This adds the second invoke (LoanAmount=50001) to the same test project. When you are done there should be two test cases listed in your test suite as shown below:

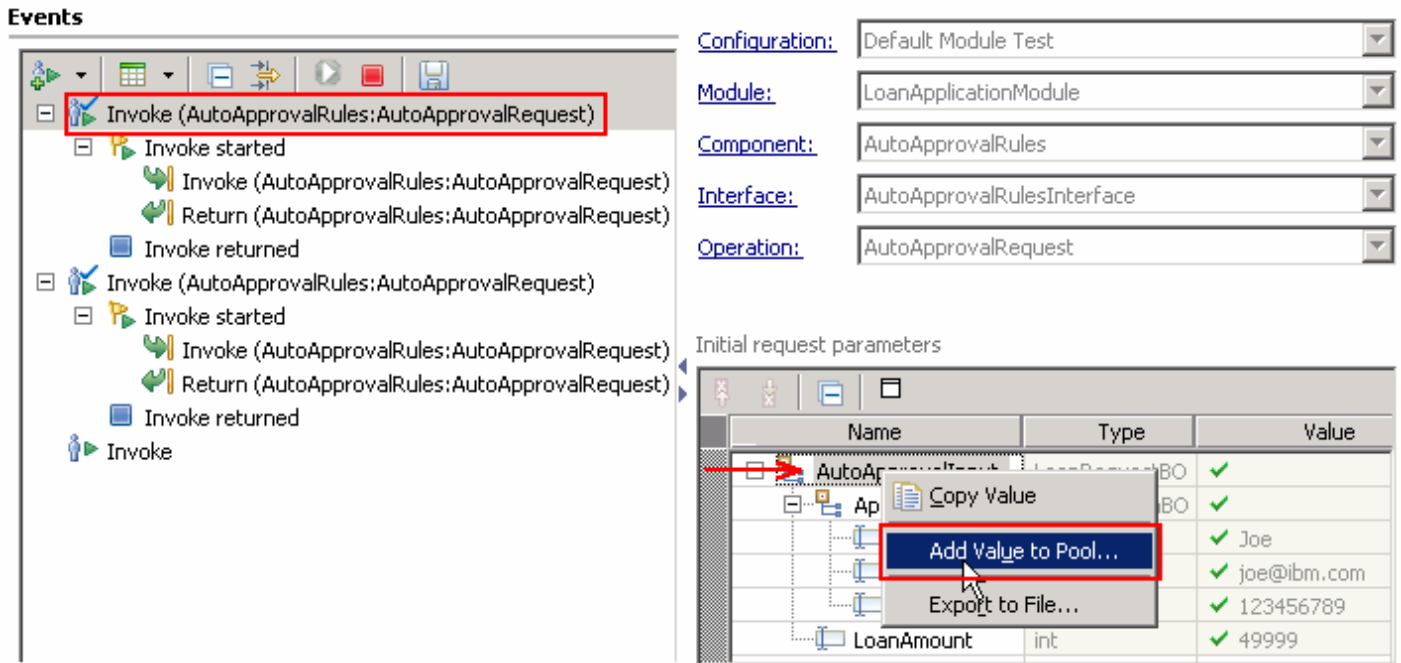


- ___ g. Save and close the **Test Cases** editor. The second test case should now appear in the business integration view:

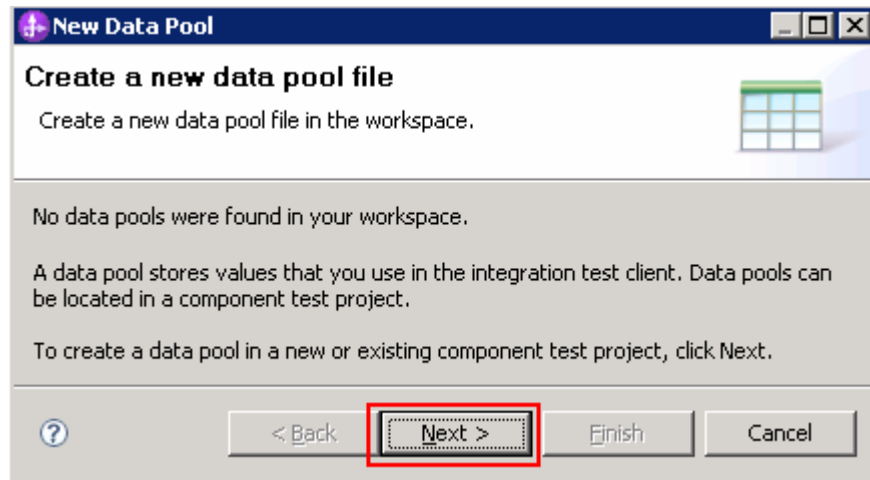


- ___ 2. Add the test values to the data pool. Using the integration test client, you can add values to data pools and then later reuse the values in the value editor.

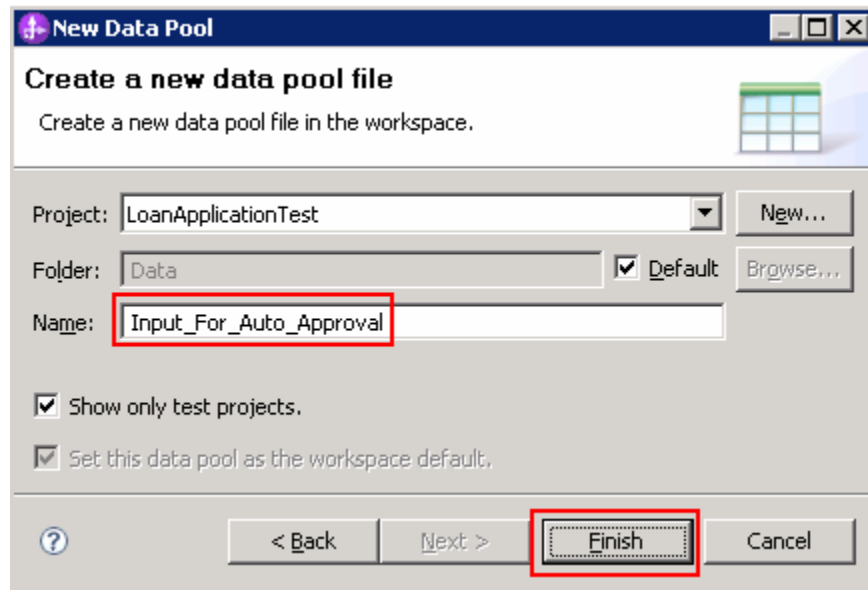
- ___ a. Add your first input parameter to the data pool. Select the first **Invoke** under **Events** and then right click over **AutoApprovalInput** under **Initial request parameters** and then select **Add Value to Pool..** as shown below:



- ___ b. The **New Data Pool** dialog is launched

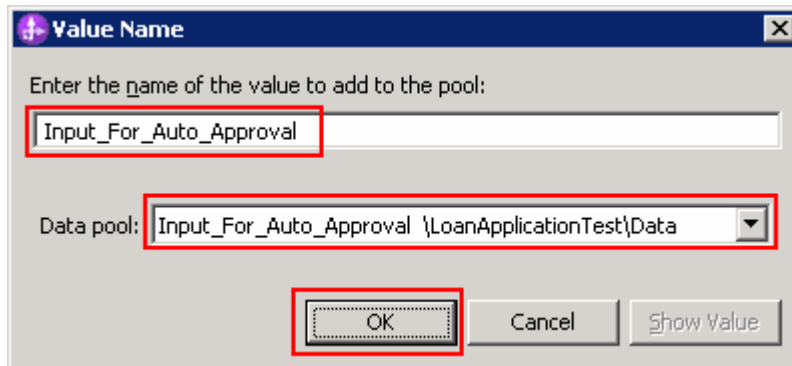


- ___ c. Click **Next**
- ___ d. In the next screen, name the data pool as **Input_For_Auto_Approval**



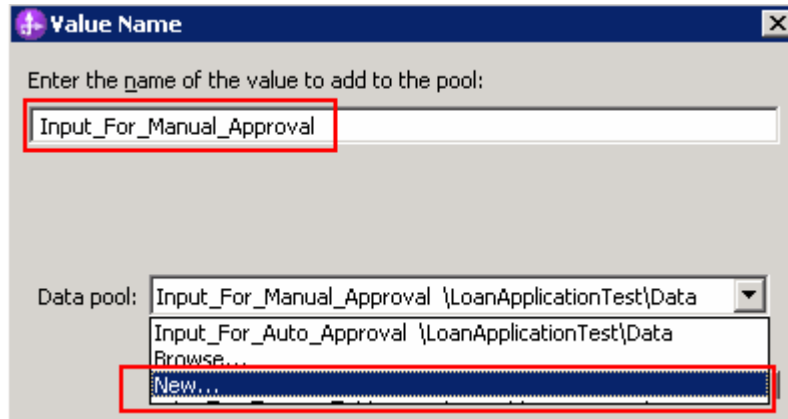
___ e. Click **Finish**

___ f. Enter the name of the value to add to the pool. By default the wizard defaults to the name of the test value element you selected to save. Type **Input_For_Auto_Approval** for the name. Accept the default for **Data pool** or select **New** to create new one.

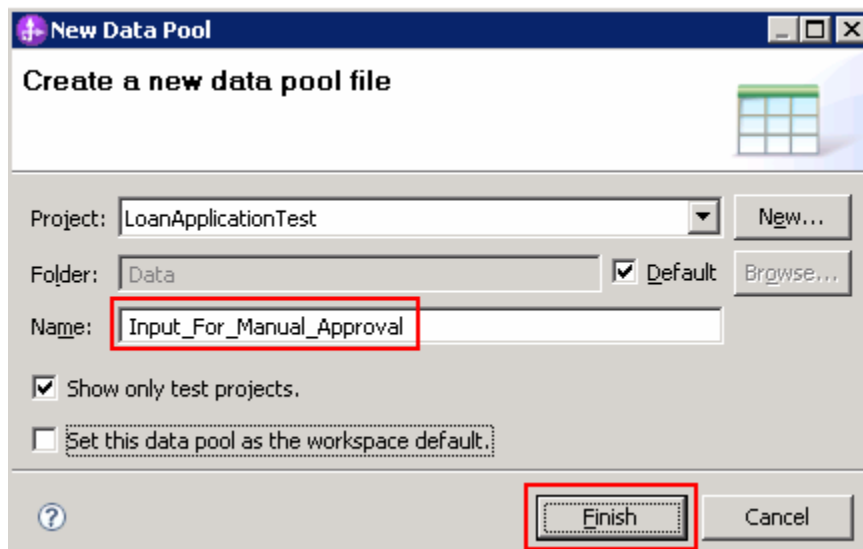


___ g. Click **OK**

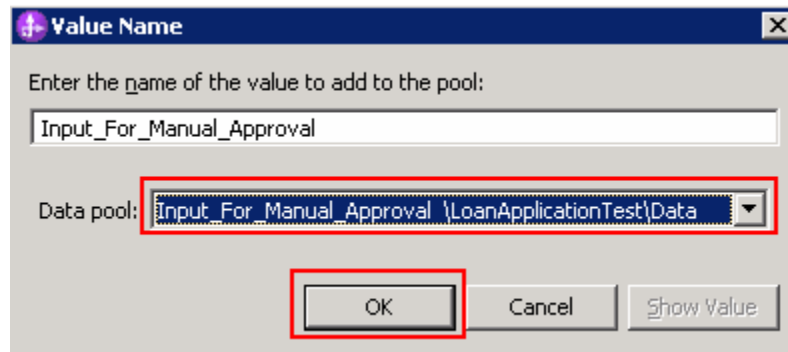
___ h. Repeat the above process to add the second input parameter to the data pool. Name it as **Input_For_Manual_Approval**. Note that while saving this test data, the test client defaults to add it to the existing data pool. Select **New** from the **Data pool** drop down



__ i. Enter **Input_For_Manual_Approval** for the name



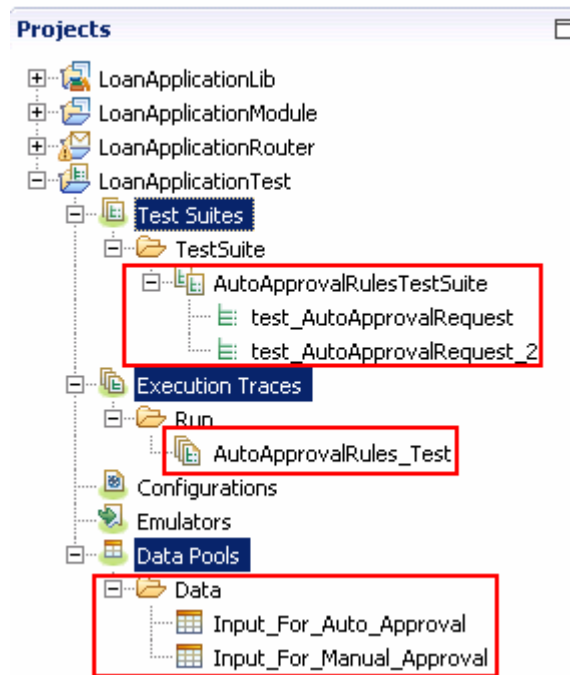
__ j. Click **Finish**. Ensure the new Data pool is selected



__ k. Click **OK**

__ l. Save the test trace and **Close** all editors.

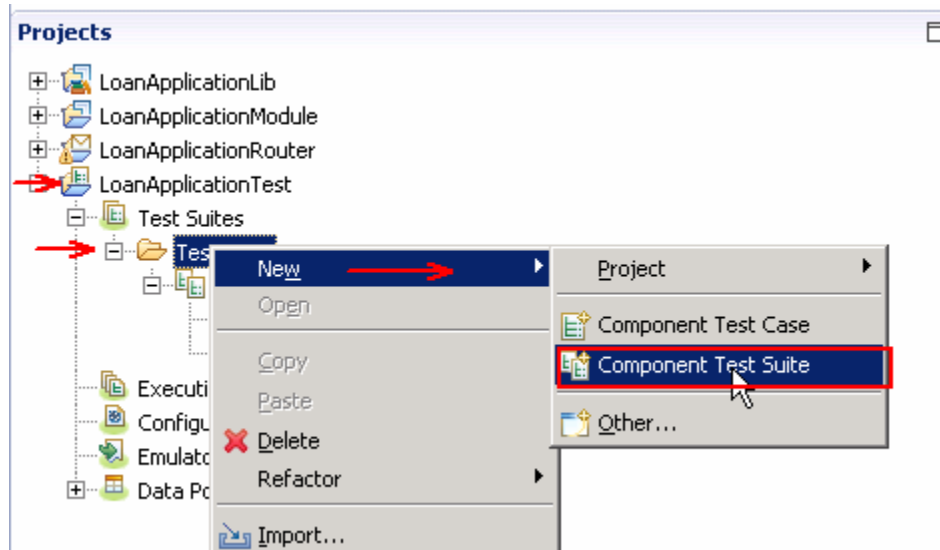
3. Check to see if the **Data Pools** are listed in the Business Integration view. Expand **LoanApplicationTest** → **Data Pools** → **Data**. Also expand **Execution Traces** → **Run**



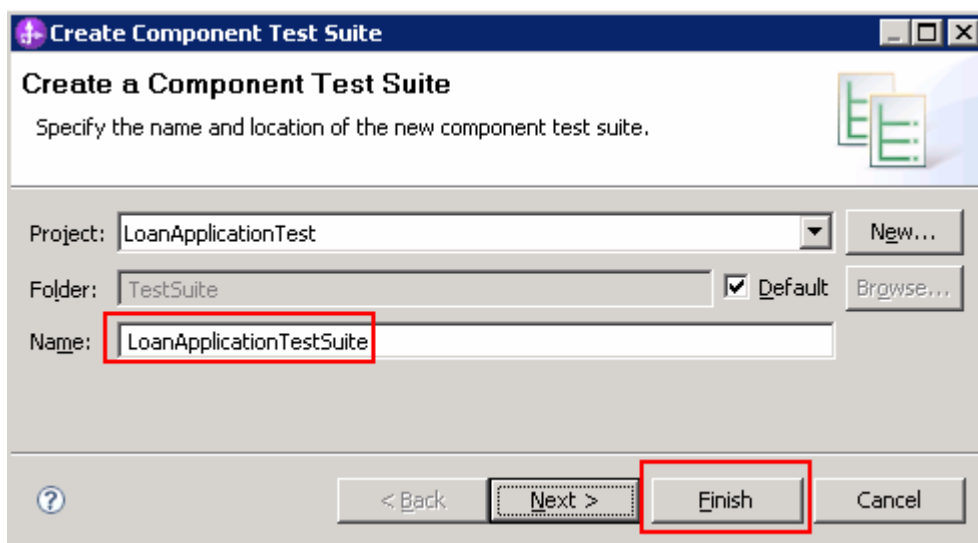
Part 4: Testing the entire application

In this part of the lab, you will add a new test case which tests the entire application. You will use the existing test artifacts.

- ___ 1. Create a new test suite
 - ___ a. In the Business Integration view, expand **LoanApplicationTest** → **Test Suites** and then right-click the **TestSuite** folder. Select **New** → **Component Test Suite** from the menu

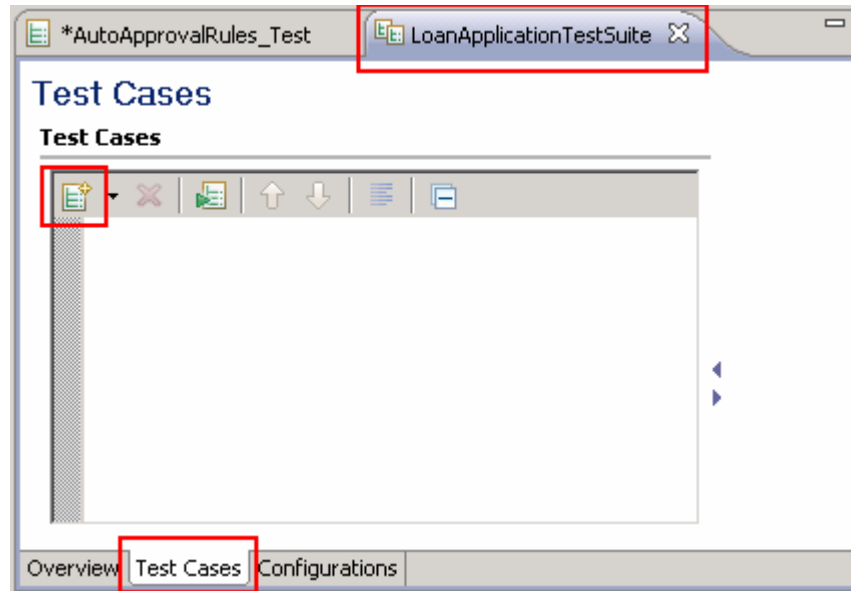



- ___ b. In the **Create Component Test Suite** enter **LoanApplicationTestSuite** for Name

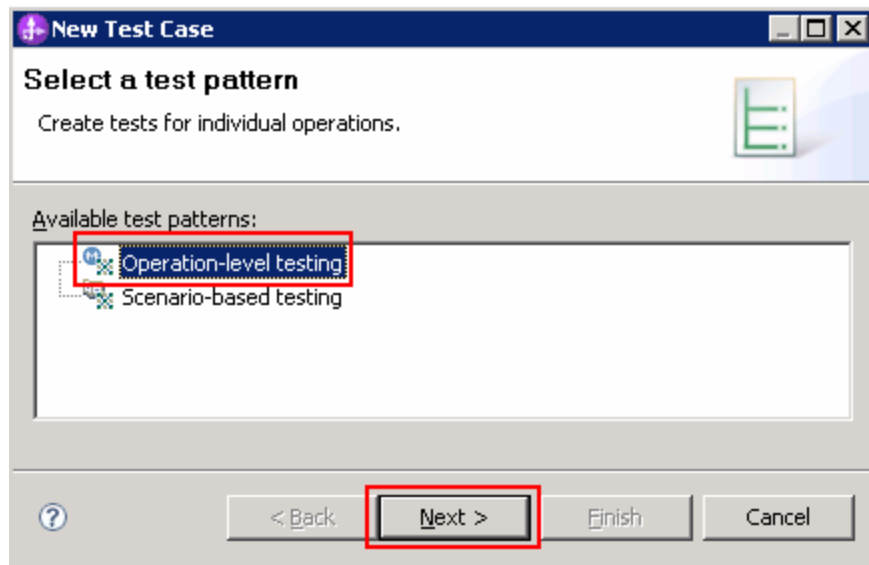


- ___ c. Click **Finish**
- ___ d. Click **Yes** to open the editor

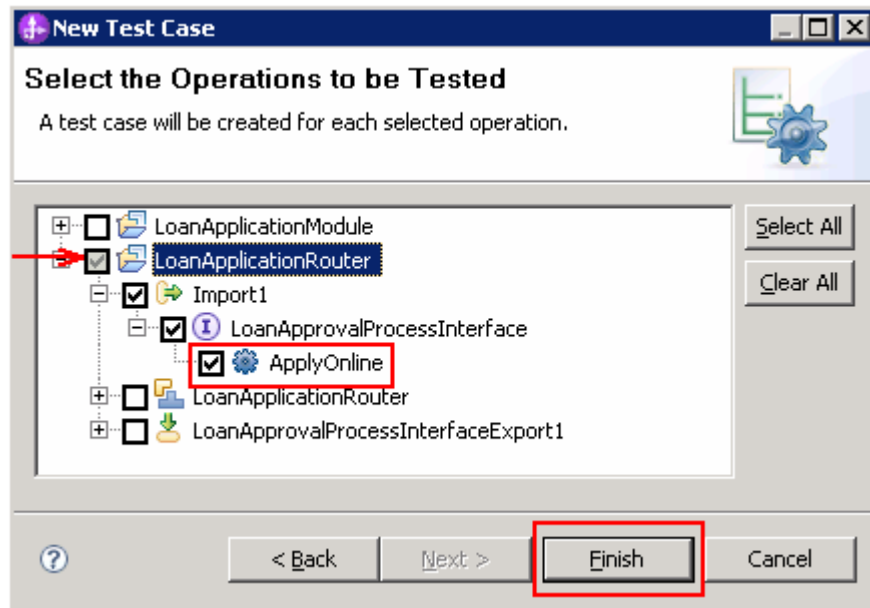
- ___ 2. Add a test case
 - ___ a. Select the **Test Cases** tab on the bottom of the editor



- ___ b. Click the **Add Test Case** button: ()
- ___ c. From the **New Test Case** dialog, select **Operation-level testing** as a test pattern. This means a test case will be automatically generated for each operation that you select and the test case makes a single invocation on the component. For scenario-based testing, a single test case is automatically generated for all of the operations that you select. The test case makes a series of invocations on one or more components that make up the scenario. You can choose the order in which you want the operations to be tested and the operations will be tested in sequence. For this lab's purposes, you will use **Operation-level testing** you want to test one operation.



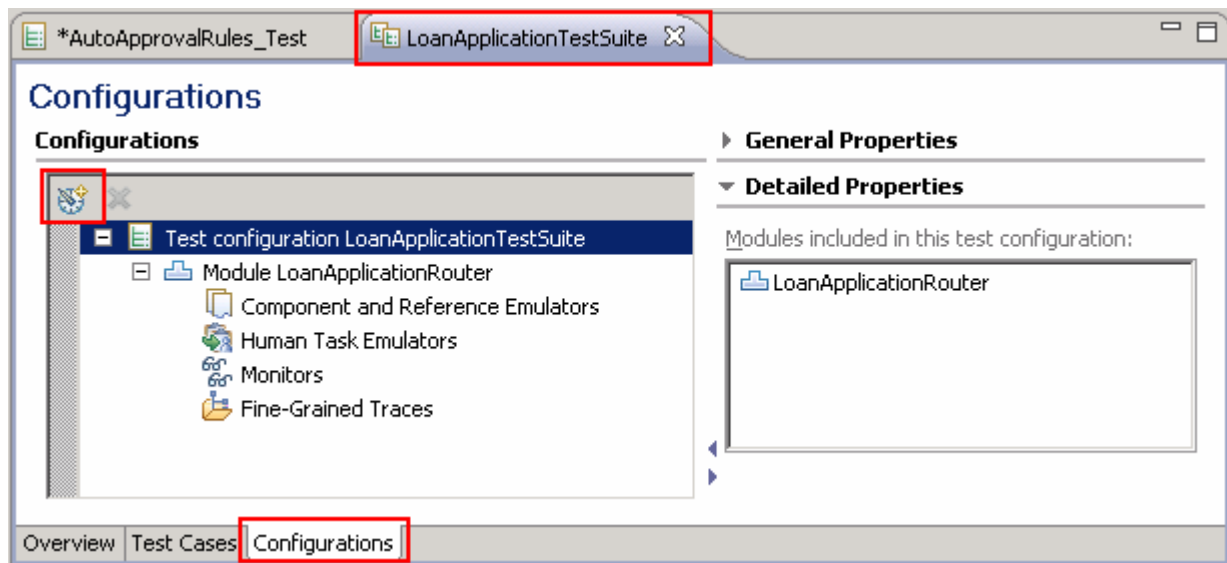
- ___ d. Click **Next**
- ___ e. In the next screen, select the operation to be tested. Expand **LoanApplicationRouter** → **Import1** → **LoanApprovalProcessInterface**. Select the check box for **ApplyOnline**



- ___ f. Click **Finish**.

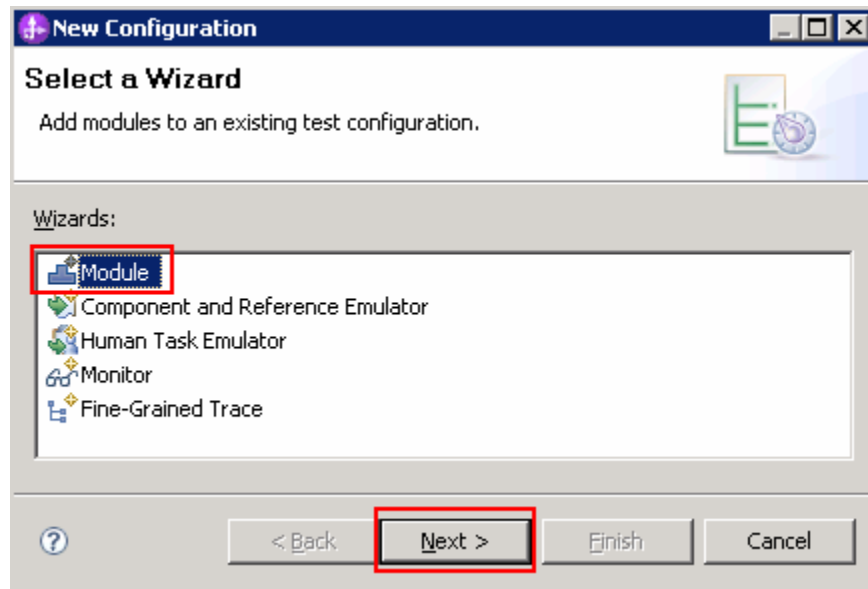
___ 3. Configure the test suite for fine-grained trace

- ___ a. Select the **Configurations** tab



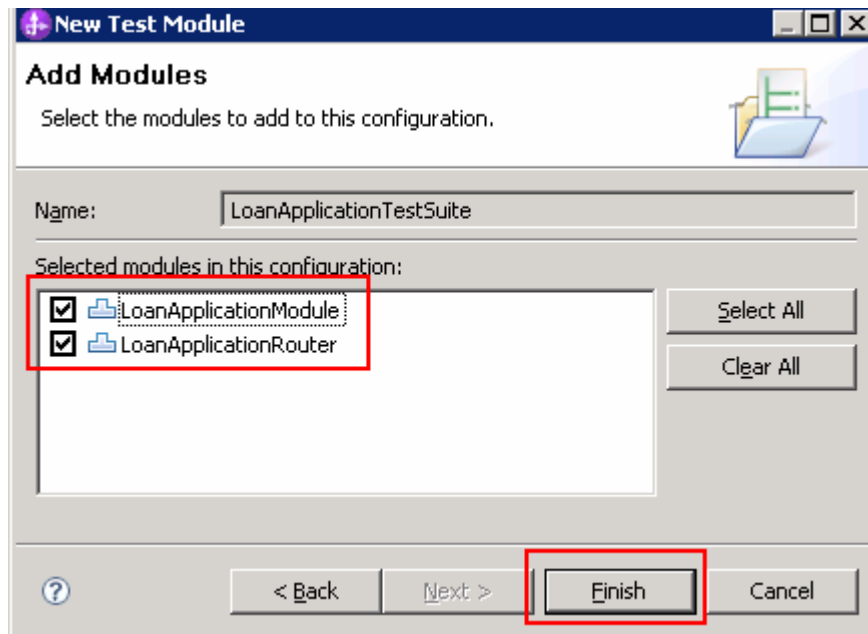
- ___ b. Click the **Add** button: ()

___ c. In the **New Configuration** dialog, select **Module** as the **Wizard**



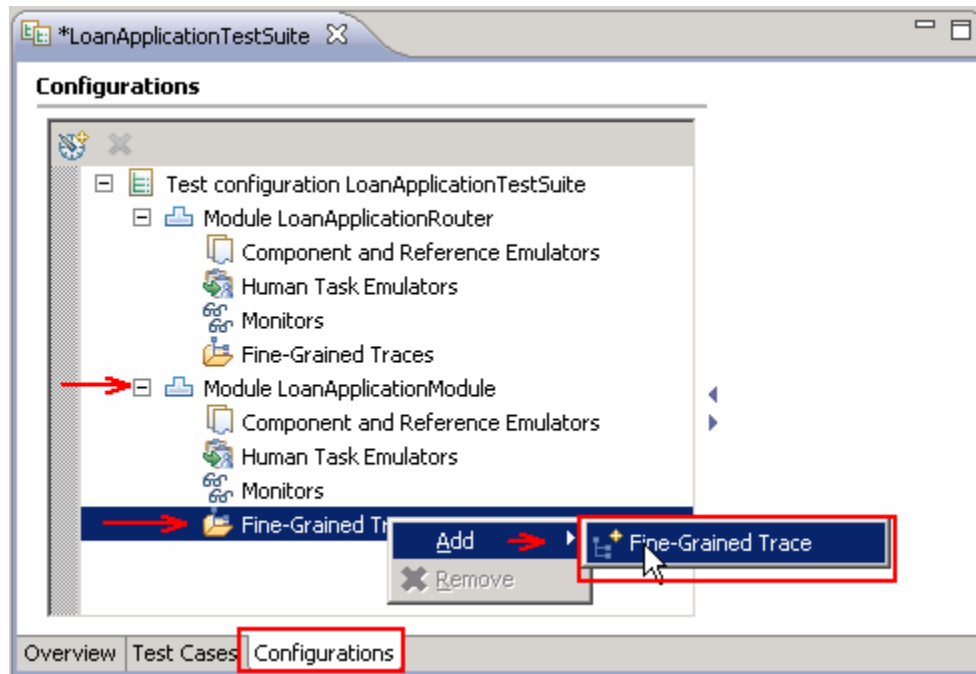
___ d. Click **Next**. Click **Next** again.

___ e. Select check box **LoanApplicationModule**

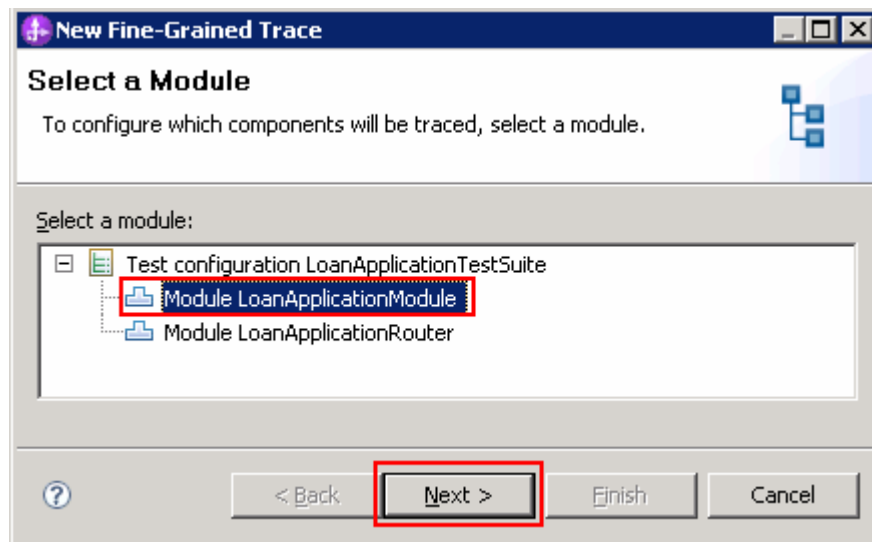


___ f. Click **Finish**

___ g. Now expand, **Module LoanApplicationModule**, right-click **Fine-Grained Traces**. Select **Add → Fine-Grained Trace** from the menu

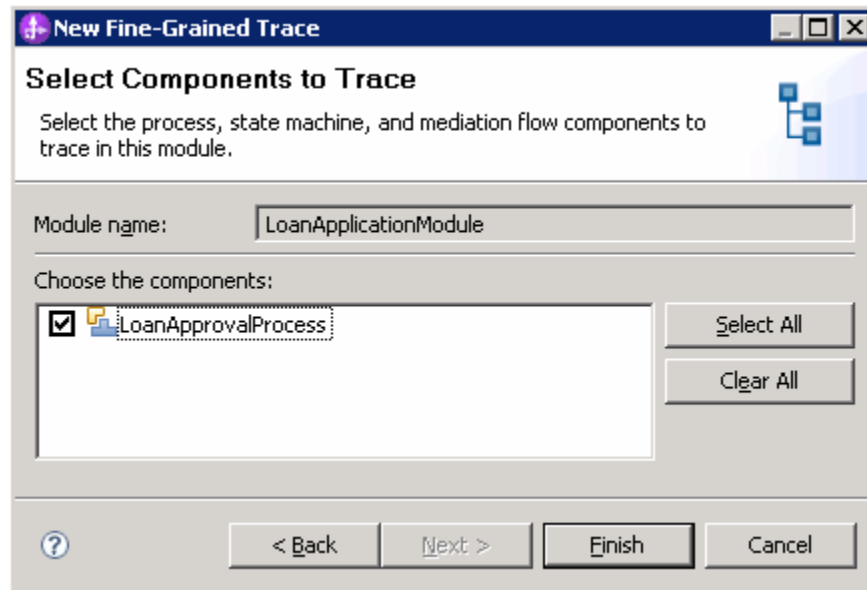


__ h. From the **New Fine-Grained Trace** dialog, select **LoanApplicationModule**



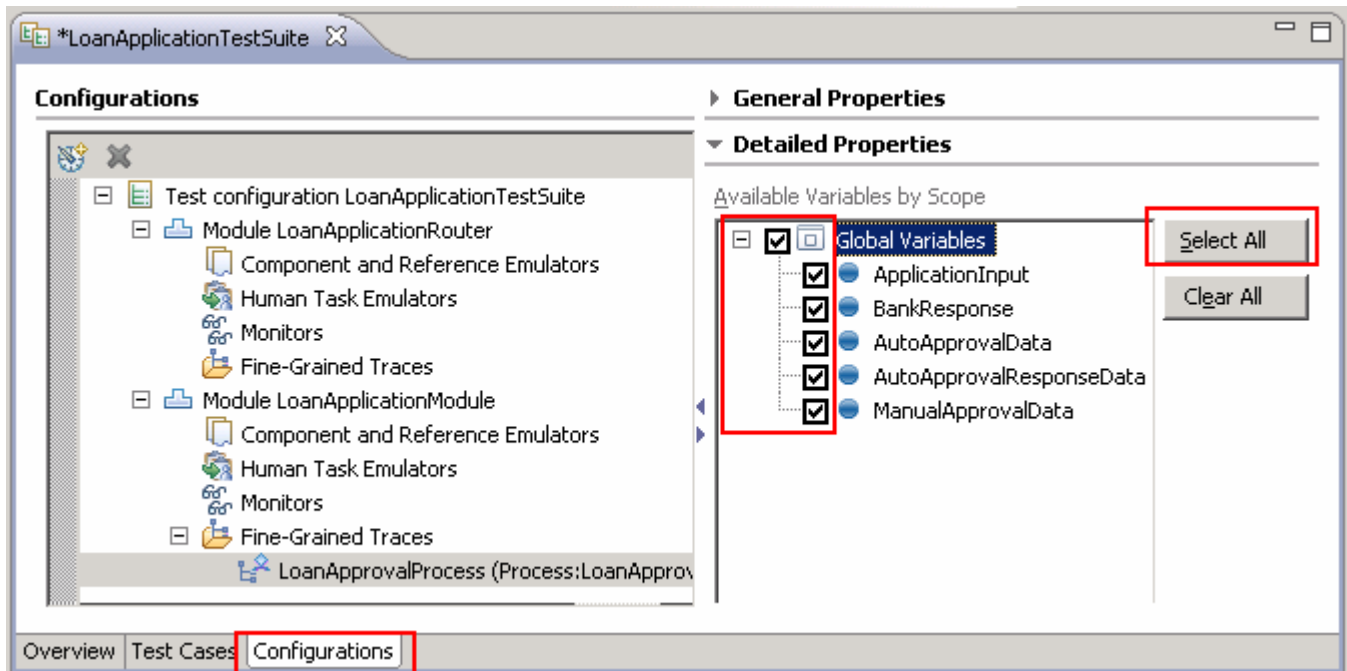
__ i. Click **Next**

___ j. In the next screen, select the **LoanApprovalProcess** check box



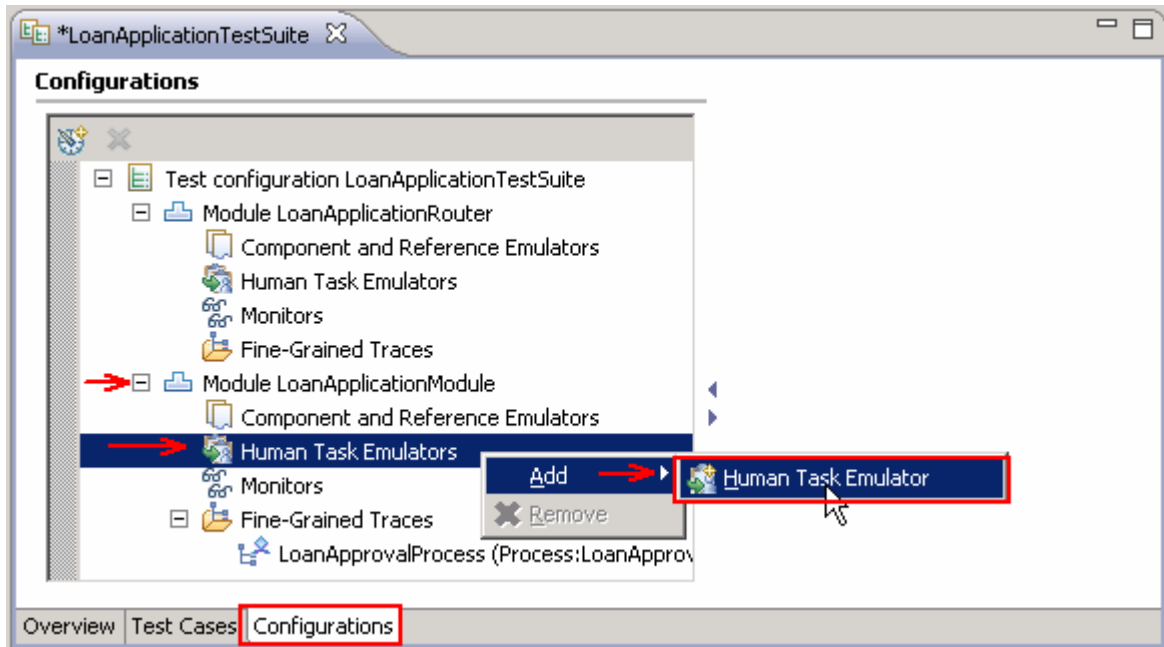
___ k. Click **Finish**.

___ l. In the Test Suite editor, while the **Configuration** tab is selected, click **Select All** on the right of the editor. This action selects all the available variables

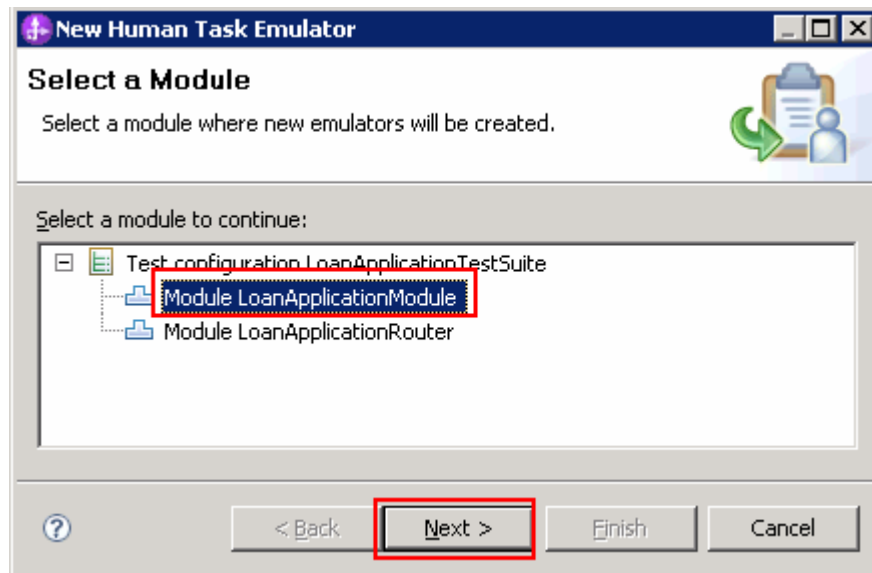


___ 4. Configure the test case for human task emulation. Human task emulation was new to v6.1.2.

- ___ a. Expand, **Module LoanApplicationModule**, right-click **Human Task Emulators**. Select **Add → Human Task Emulator** from the menu

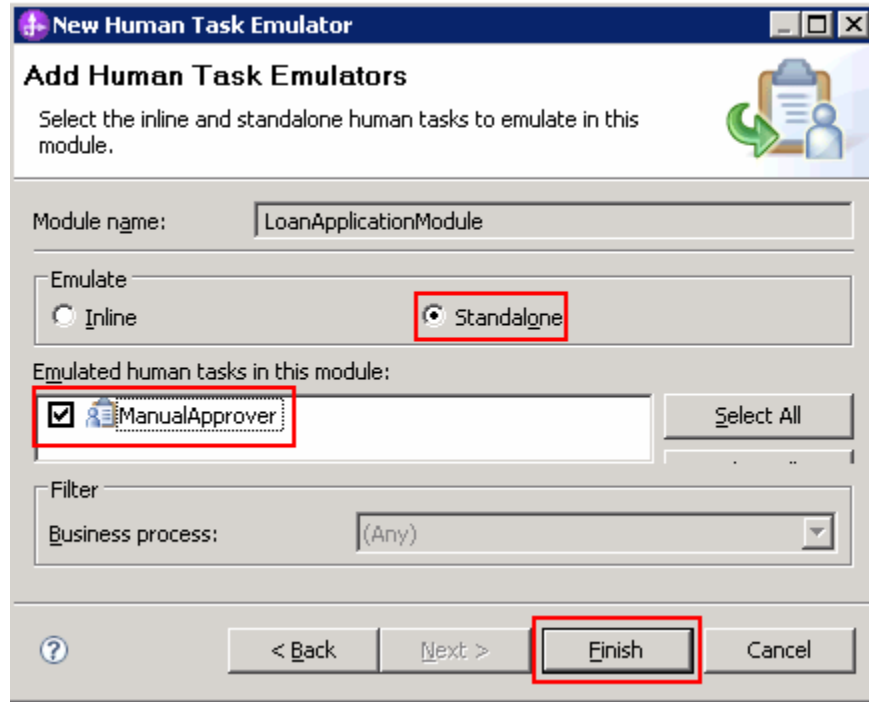


- ___ b. In the **New Human Task Emulator**, ensure **Module LoanApplicationModule** is selected.

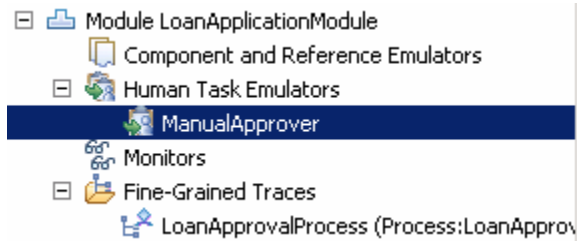


- ___ c. Click **Next**

___ d. In the next screen, select **Standalone** radio button and then select the check box for **ManualApprover**

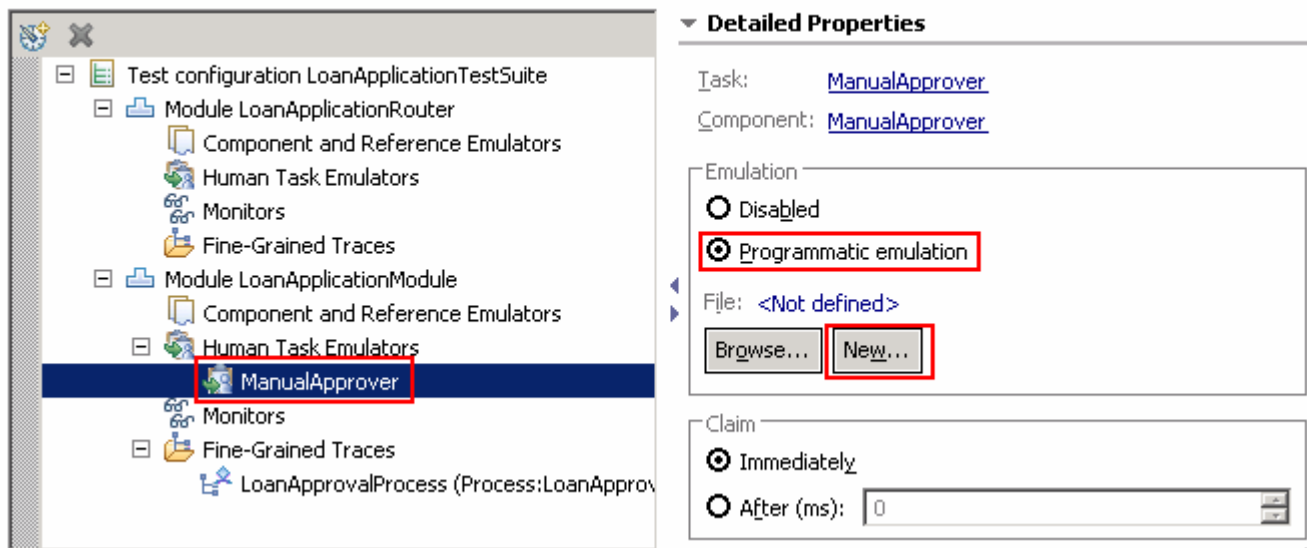


___ e. Click **Finish**. **ManualApprover** appears under **Human Task Emulators**:



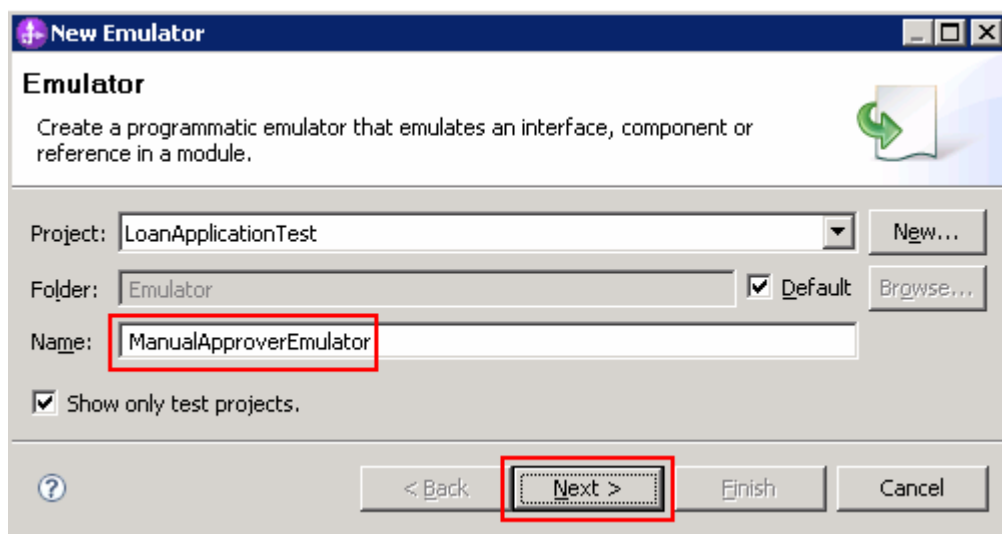
___ f. Now, select **ManualApprover** and then select **Programmatic emulation** radio button on the right side under Detailed Properties section

Configurations



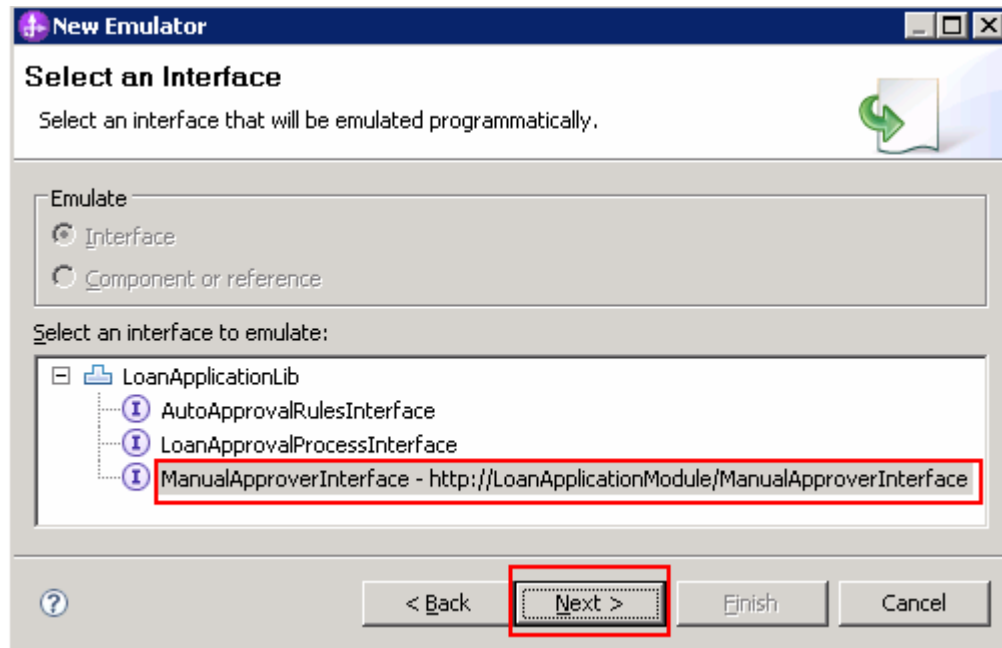
___ g. Click **New**. The **New Emulator** dialog is launched

___ h. Name the emulation as **ManualApproverEmulator**



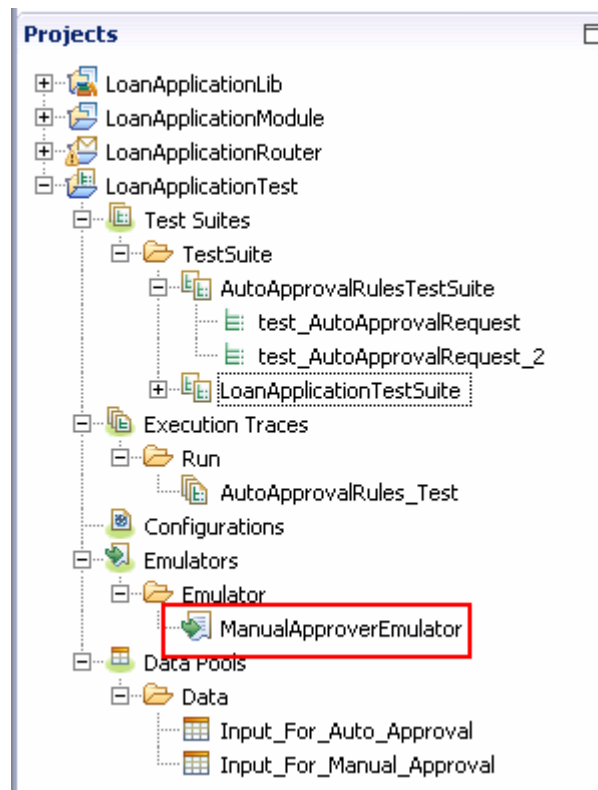
___ i. Click **Next**

___ j. In the next screen, select the **ManualApprovalInterface** to emulate

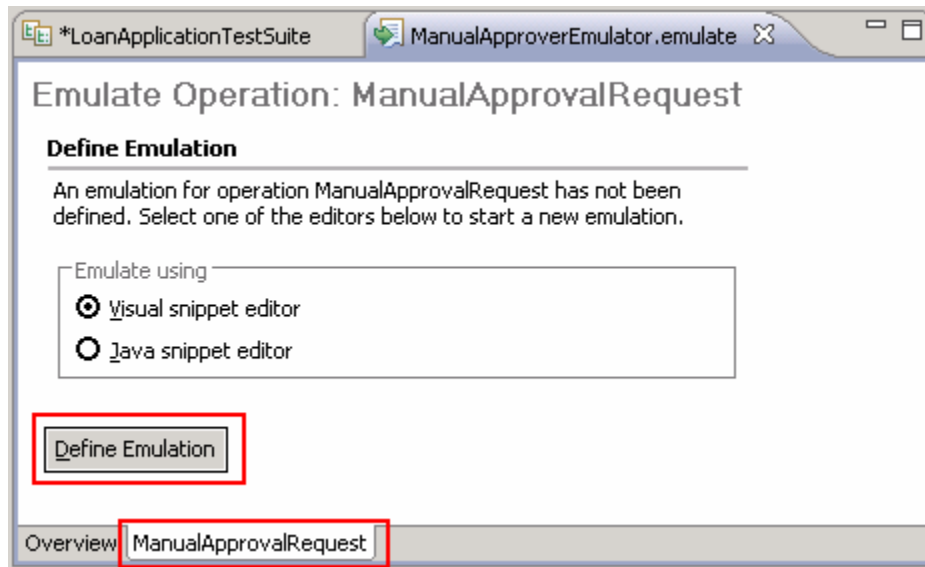


___ k. Click **Next**

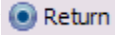
___ l. Use the default Java class and click **Finish**. The emulation opens in the emulation editor. Also notice that it appears in the **Business Integration** view as shown below:

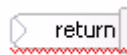


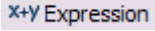
___ m. In the emulation editor, select the **ManualApprovalRequest** tab at the bottom.

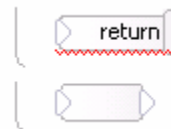


___ n. Ensure the **Visual snippet editor** radio button is selected and then click the **Define Emulation** button. This opens the visual snippet editor.

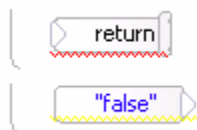
___ o. From the palette of the visual snippet editor, click the **Return** button: () and then drop it on the canvas by clicking the white space



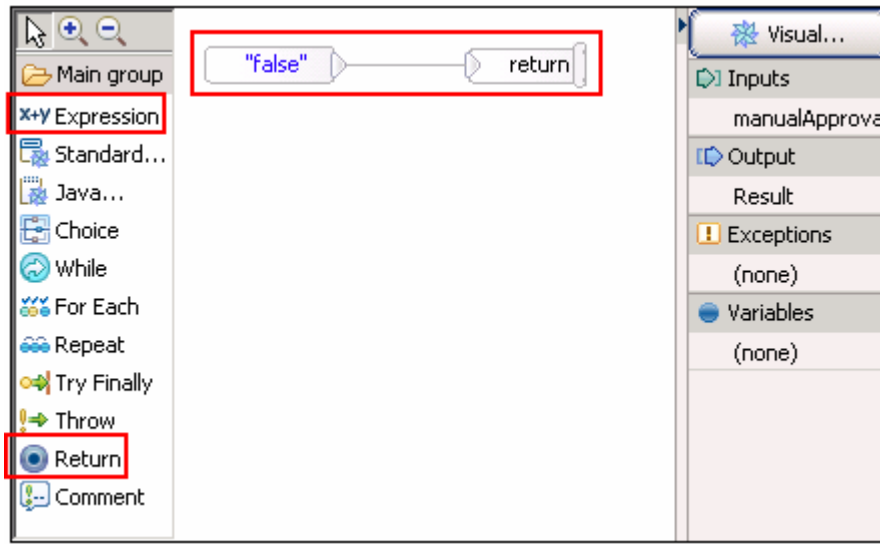
___ p. Now, click the **Expression** button from the palette: () and then drop it over the canvas by clicking on the white space



___ q. Enter a value of **"false"**. Include the quotation marks because it is a string.



___ r. Wire the expression to the return activity. Your visual snippet should look like as shown below:



___ s. Save and close the visual snippet editor

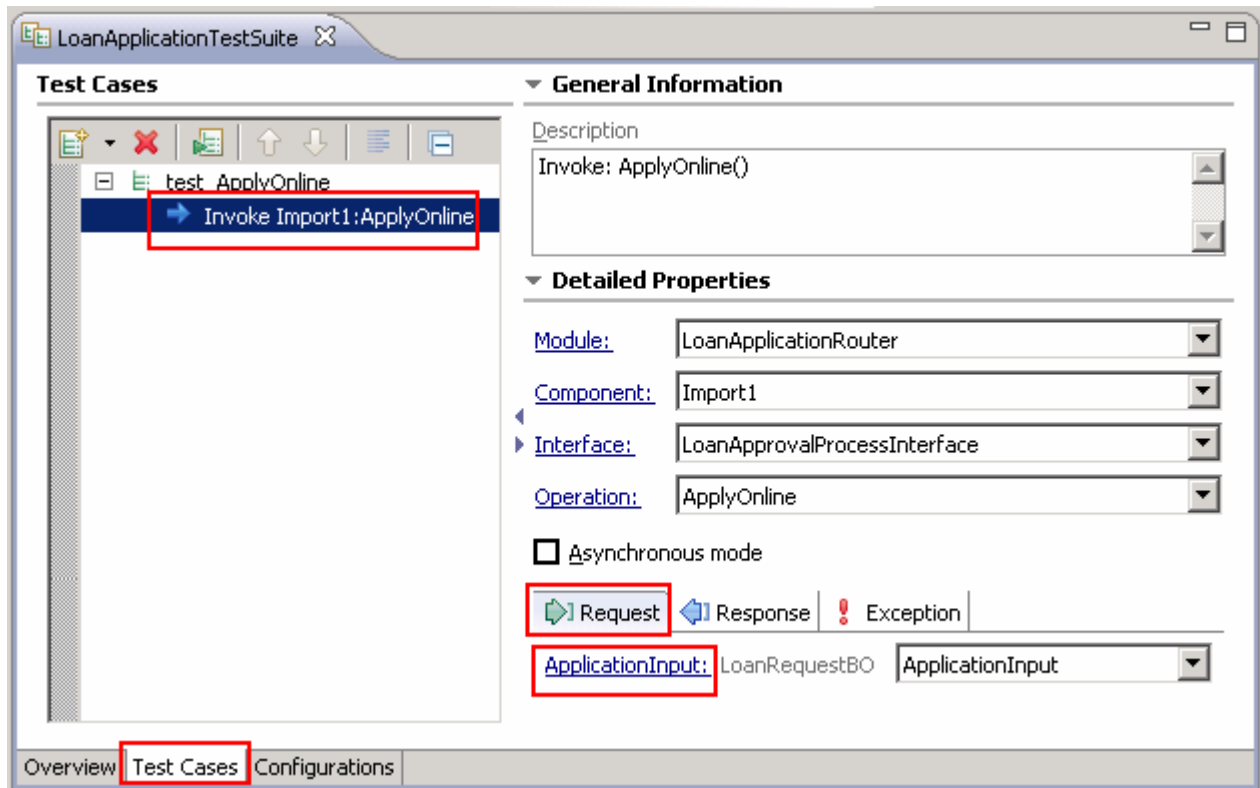
___ t. The Test Suite editor for **LoanApplicationTestSuite** should still be open. On the right, note that you can choose to claim the human task immediately, or wait for a number of milliseconds. Also note that you can assign a potential owner who claims the task. You will leave both as default.

___ u. Save editor

___ 5. Configure the input and output parameters

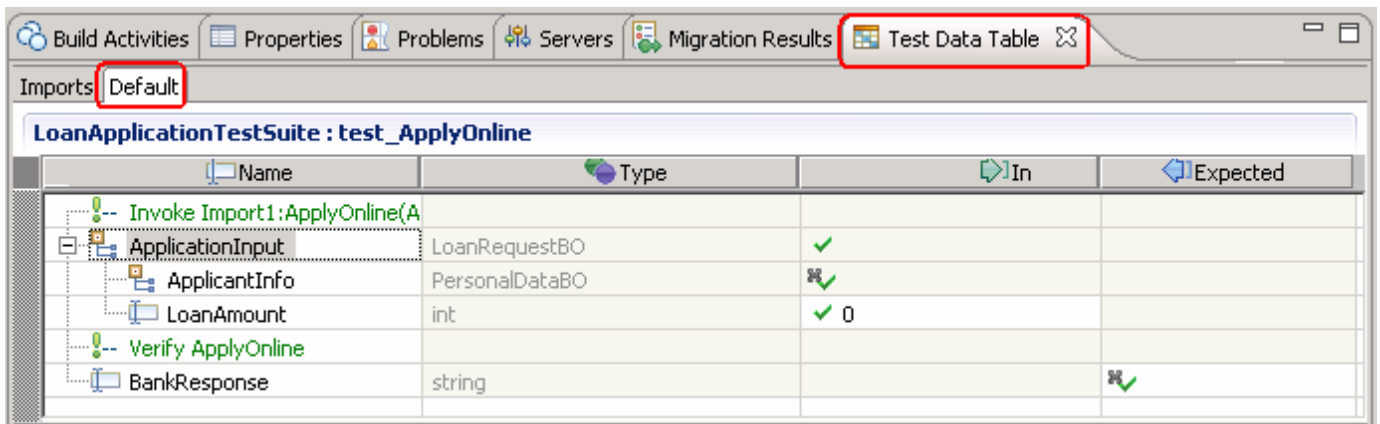
___ a. Select the **Test Cases** tab at the bottom of the editor.

___ b. Ensure **Invoke Import1:ApplyOnline** operation is selected in the editor.

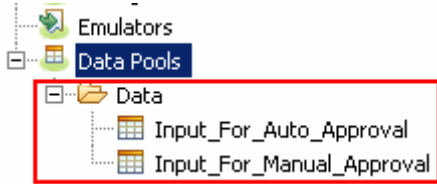


___ c. Now select the **Request** tab and then click **ApplicationInput** under the Detailed Properties section:

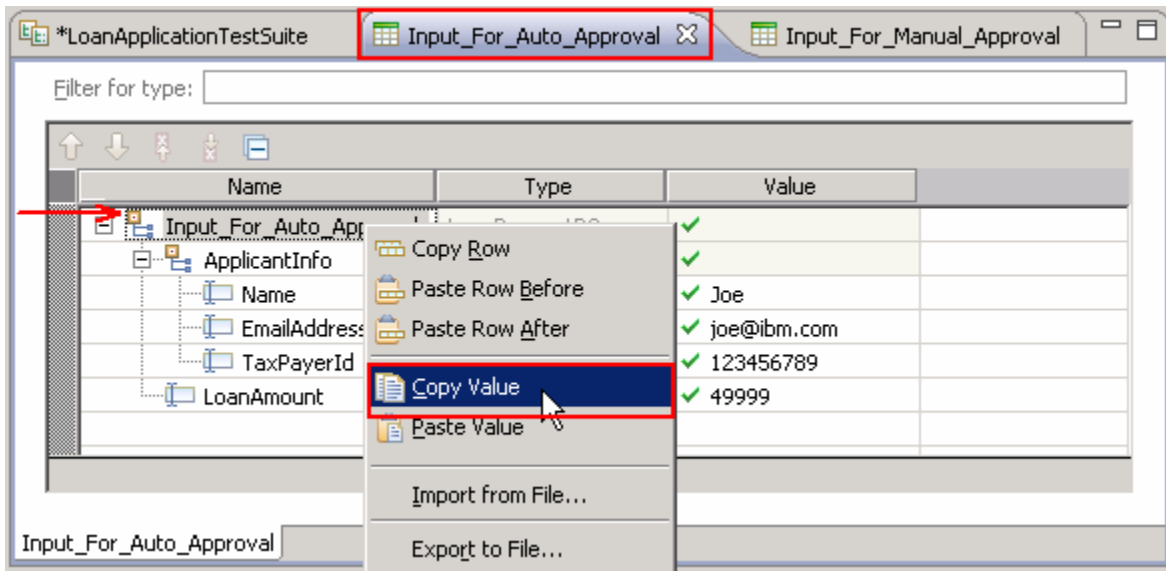
___ d. Note the **Test Data Table** view opens in the frame below



- ___ e. In the **Business Integration** view, expand **LoanApplicationTest** → **Data Pools** → **Data**. Double click to open both the **Input_For_Auto_Approval** and **Input_For_Manual_Approval** data pools

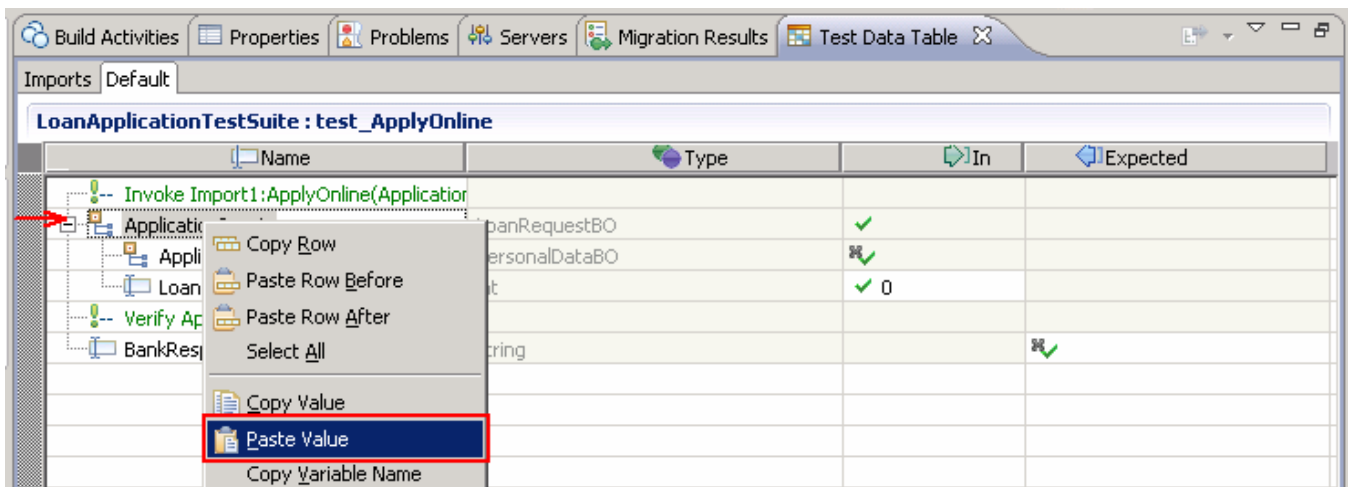


- ___ f. In the **Data Pool** editor, right click **Input_For_Auto_Approval** and select **Copy Value**



- ___ g. Close the Data Pool editor.

- ___ h. Now, in the **Test Data Table** editor, right-click **ApplicationInput** and select **Paste Value**:



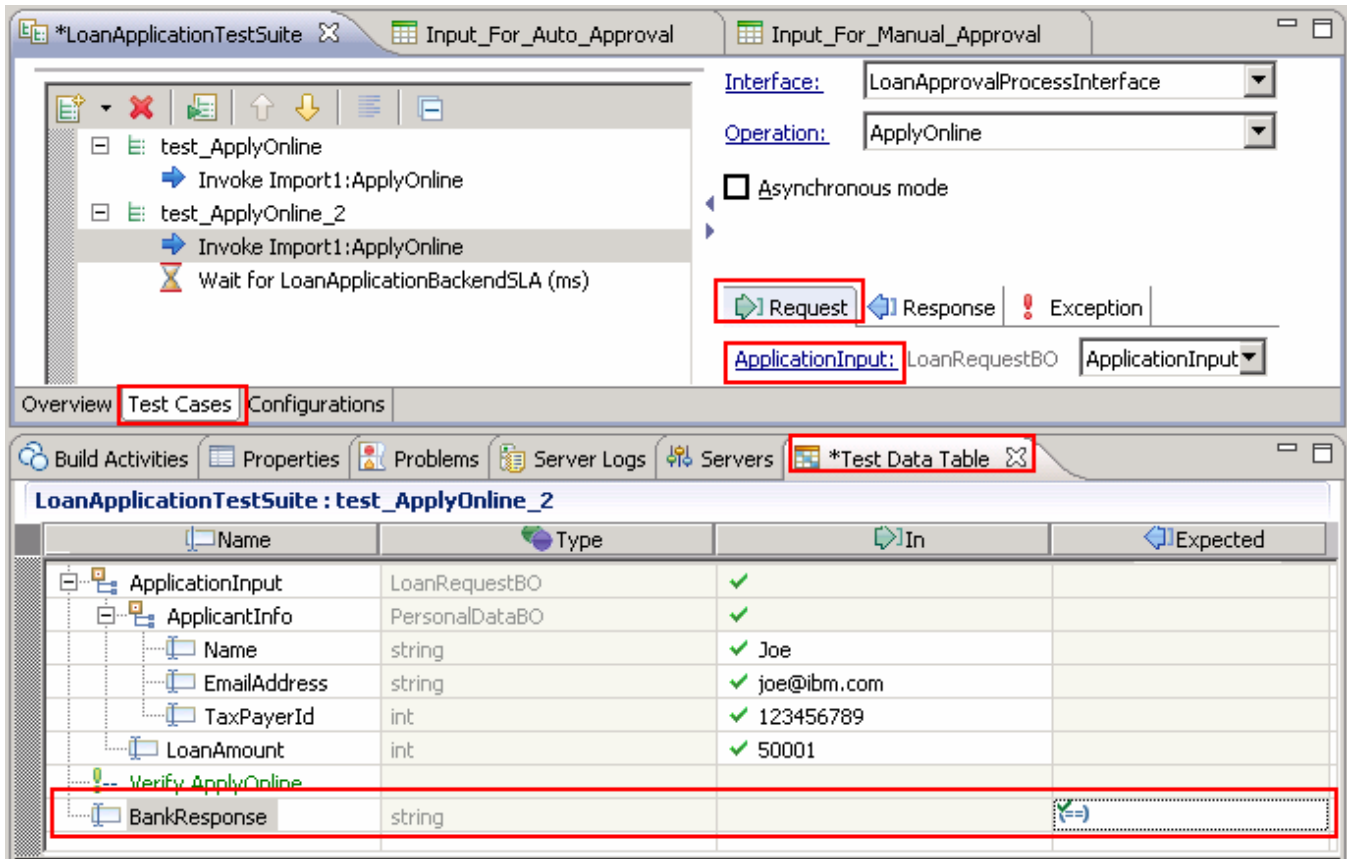
___ i. Enter "Your application has been approved." for the **BankResponse**

Note: Do not forget the period or the test will fail.

Name	Type	In	Expected
Invoke Import1:Apply			
ApplicationInput	LoanRequestBO	✓	
ApplicantInfo	PersonalDataBO	✓	
Name	string	✓ Joe	
EmailAddress	string	✓ joe@ibm.com	
TaxPayerId	int	✓ 123456789	
LoanAmount	int	✓ 49999	
Verify ApplyOnline			
BankResponse	string		✓ Your application has been approved.

___ j. Save the Test Data Table

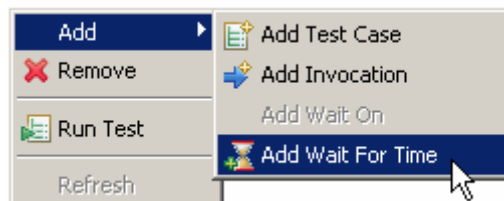
- ___ k. Similarly add another test case for a **LoanAmount** of 50001. Remember, the response for this case is **false** since the process returns the result from the human task. Instead of entering the value of **false**, leave the field blank to learn what happens when a test case fails.



- ___ l. Save the Test Data table

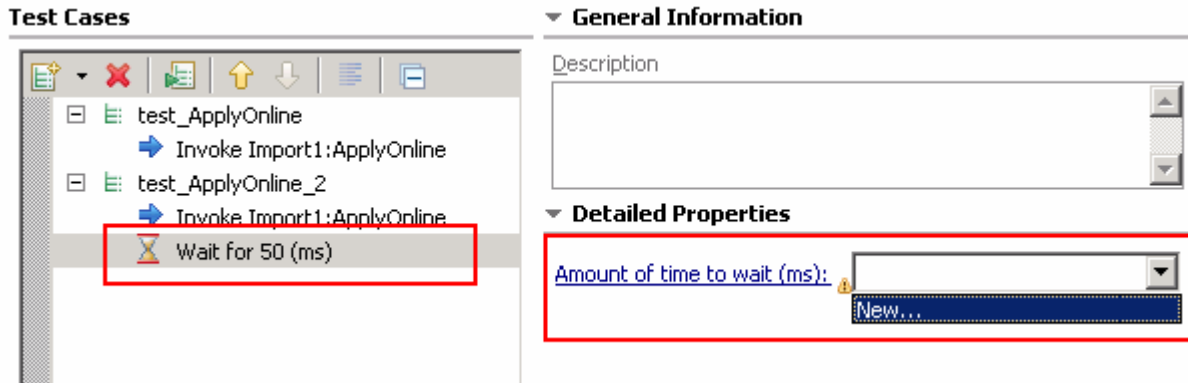
- ___ 6. Add a **Wait For Time** to the component test. A wait for time will pause testing for a certain amount of milliseconds so you can re-create a real-life scenario or purposely make a response time out so an error is thrown. You can also pause between modules. The wait works by creating a variable that you name to hold a value of type long which represents time in milliseconds.

- ___ a. Right click the second Test Case (**test_ApplyOnline_2**) and select **Add → Add Wait For Time**.

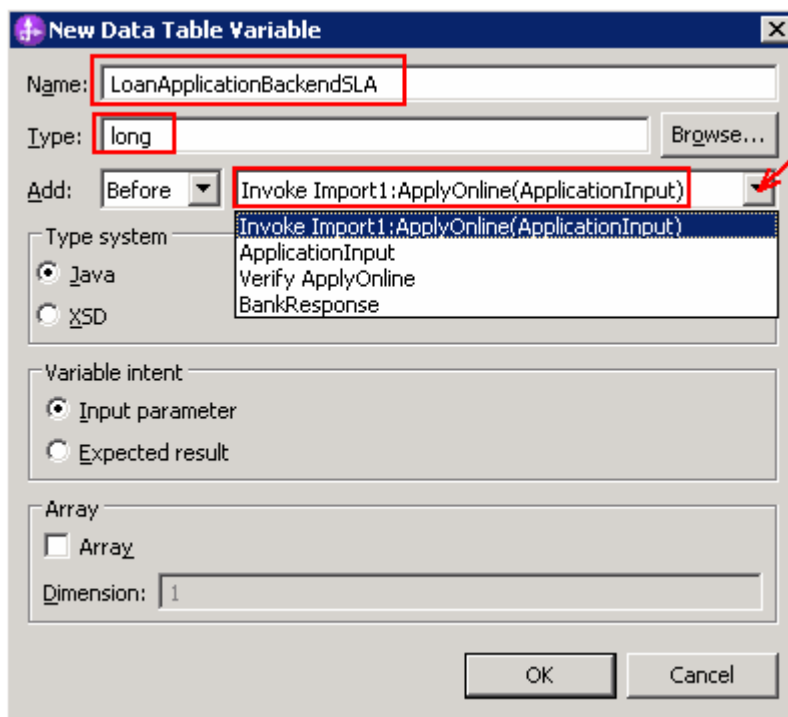


- ___ b. A default wait variable is created for 50 ms (milliseconds). Create a new wait variable

- 1) Click the drop down box for **Amount of time to wait (ms)** and select **New...**



- 2) In the New Data Table Variable dialog, add the name **LoanApplicationBackendSLA**. Notice the options here for a data table variable. However, for the wait, you will use the default of type long, and set to pause before invoke of the import.



- 3) Click **OK**
- 4) The Test Data Table view now displays the wait before the invoke of import1

LoanApplicationTestSuite : test_ApplyOnline_2

Name	Type	In	Expected
LoanApplicationBackendSL	long	✓ 0	
Invoke Import1:ApplyOnline			
ApplicationInput	LoanRequestBO	✓	
ApplicantInfo	PersonalDataBO	✓	
Name	string	✓ Joe	
EmailAddress	string	✓ joe@ibm.com	
TaxPayerId	int	✓ 1234567	
LoanAmount	int	✓ 50001	
Verify ApplyOnline			
BankResponse	string		⇐=

5) Click the 0 under the **In** column  and enter **10000**

LoanApplicationTestSuite : test_ApplyOnline_2

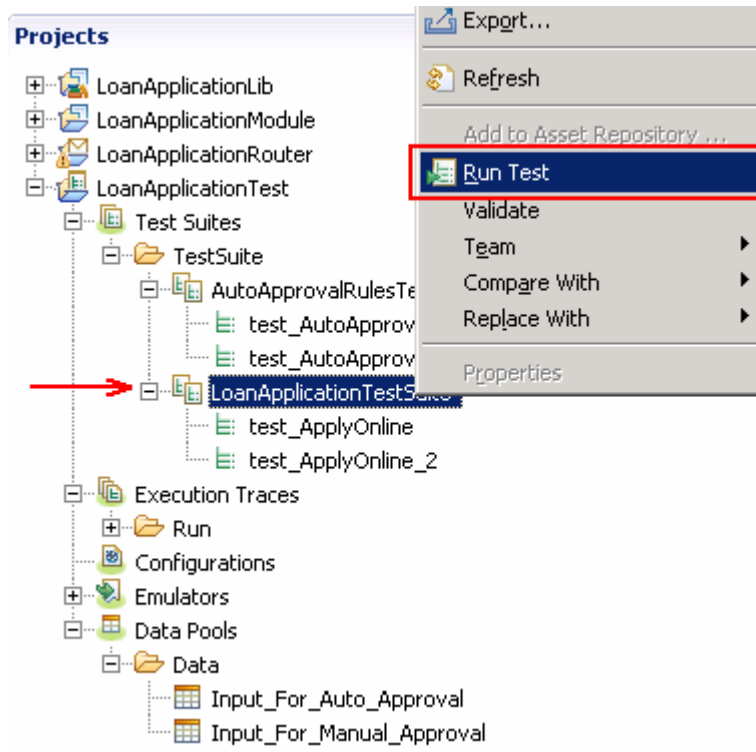
Name	Type	In	Expected
LoanApplicationBackendSL	long	✓ 10000	
Invoke Import1:ApplyOnline			


___ 7. Save the editor.

Part 5: Running a test suite

In this part of the lab, you will run the test suite.

- ___ 1. Run the **LoanApplicationTestSuite** test suite
 - ___ a. In the **Business Integration** view, right click **LoanApplicationTest** → **Test Suites** → **TestSuite** → **LoanApplicationTestSuite** and select **Run Test**



- ___ b. The test suite will open in the editor.
- ___ c. Click the **Continue** button: 
- ___ d. One test should pass and one should fail as shown below

Events

General Properties

Detailed Properties

Configuration: LoanApplicationTestSuite

Verdict: Failed

Total: 2/2

2/2

Passed: 1

1

Failed: 1

1

Error:

0

- e. Expand the test case that is marked as passed (**test_ApplyOnline**) and then select **Return** element. You should see the response as **Your application had been approved.** in the **Expected** column

Events

General Properties

Detailed Properties

Module: [LoanApplicationRouter](#)

Component: [Import1](#)

Interface: [LoanApprovalProcessInterface](#)

Operation: [ApplyOnline](#)

Return parameters:

Name	Type	Value
BankResp...	string	✓ Your application has been approved.

- ___ f. Now expand the second test case that marked as failed (test_ApplyOnline_2) and then select **Test Variation**. You should see a blank response in the **Expected** column. It failed because the message “false” was returned instead of the blank space you put in for the expected response.

The screenshot displays the Test Client interface. On the left, the 'Events' tree shows a hierarchy of test results: 'Run Test (LoanApplicationTestSuite) [Failed]' contains 'Test Suite (LoanApplicationTestSuite) [Failed]', which contains 'Test Case (test_ApplyOnline) [Passed]' and 'Test Case (test_ApplyOnline_2) [Failed]'. The 'Test Case (test_ApplyOnline_2) [Failed]' is expanded to show 'Test Variation (Default) [Failed]' and 'Invoke Import1:ApplyOnline'. A red arrow points to the failed test case, and a red box highlights the failed test variation. On the right, the 'General Properties' and 'Detailed Properties' panels are visible. The 'Detailed Properties' panel shows 'Test variation: Default' and 'Verdict: Failed'. Below, the 'Results' table is shown with a red box highlighting a row where the actual result is 'false' and the expected result is '=='. The table has columns for Name, Type, Actual, and Expected.

Name	Ty...	Actual	Expected
BankRe...	string	✓ false	(==)

- ___ g. Close the Test Client

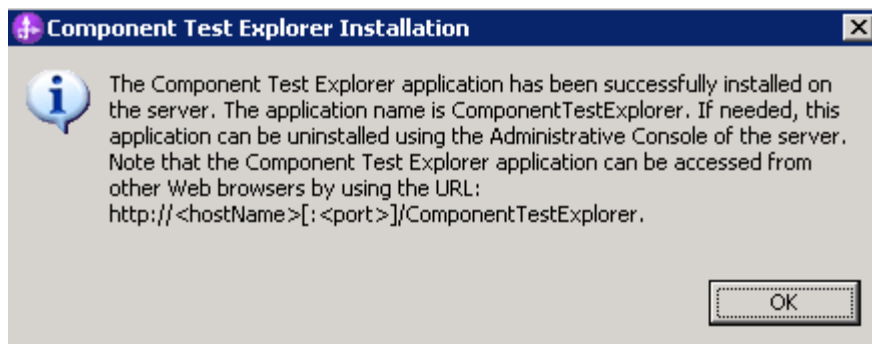
Part 6: Using the component test explorer

Component test explorer enables you to manage and run test cases that have been deployed to either a WebSphere test environment server or a stand-alone server. Much like the Business Process Choreographer Explorer, the Component Test Explorer is a web client that you can invoke and run from either inside or outside of WebSphere Integration Developer.

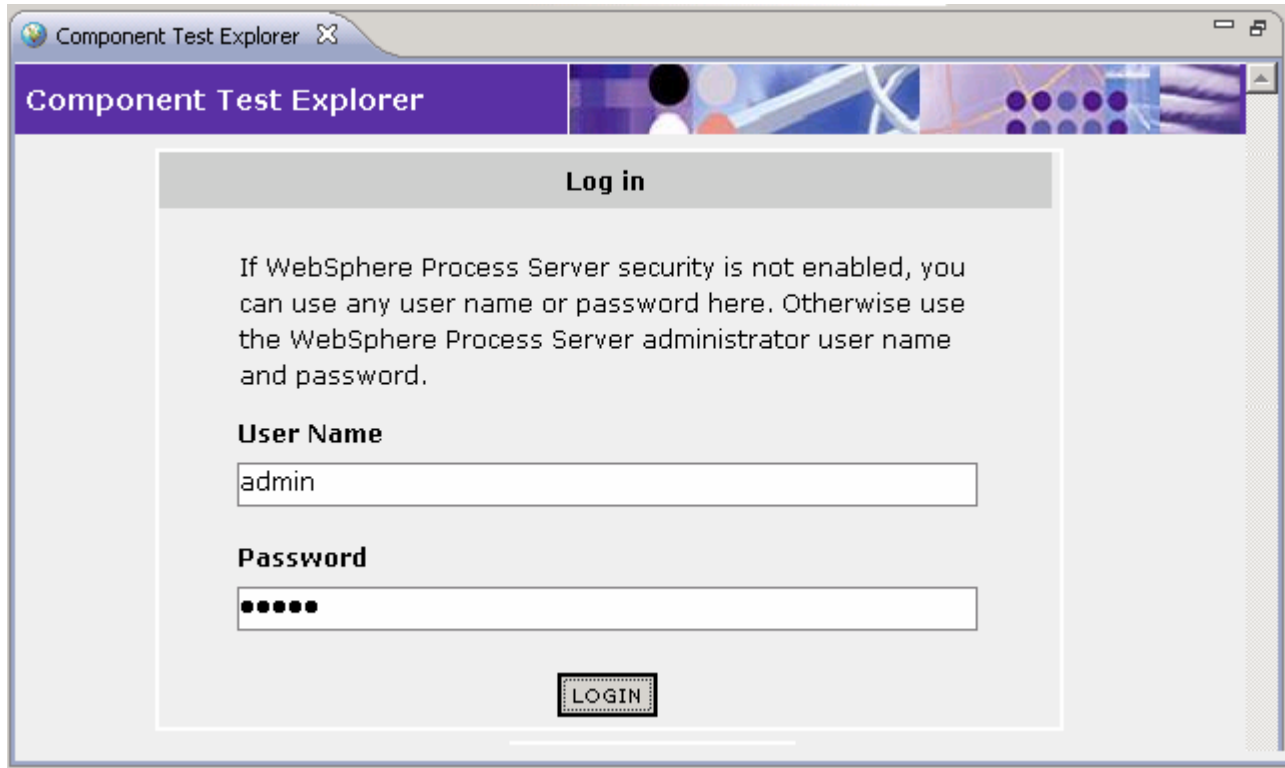
- ___ 1. Launch the component test explorer
 - ___ a. In the **Servers** view, right-click the running server and select **Launch → Component Test Explorer**
 - ___ b. By default the **Component Test Explorer** application is not installed to the test server. If this is the first time you launched the Component Test Explorer, you will be prompted for installation



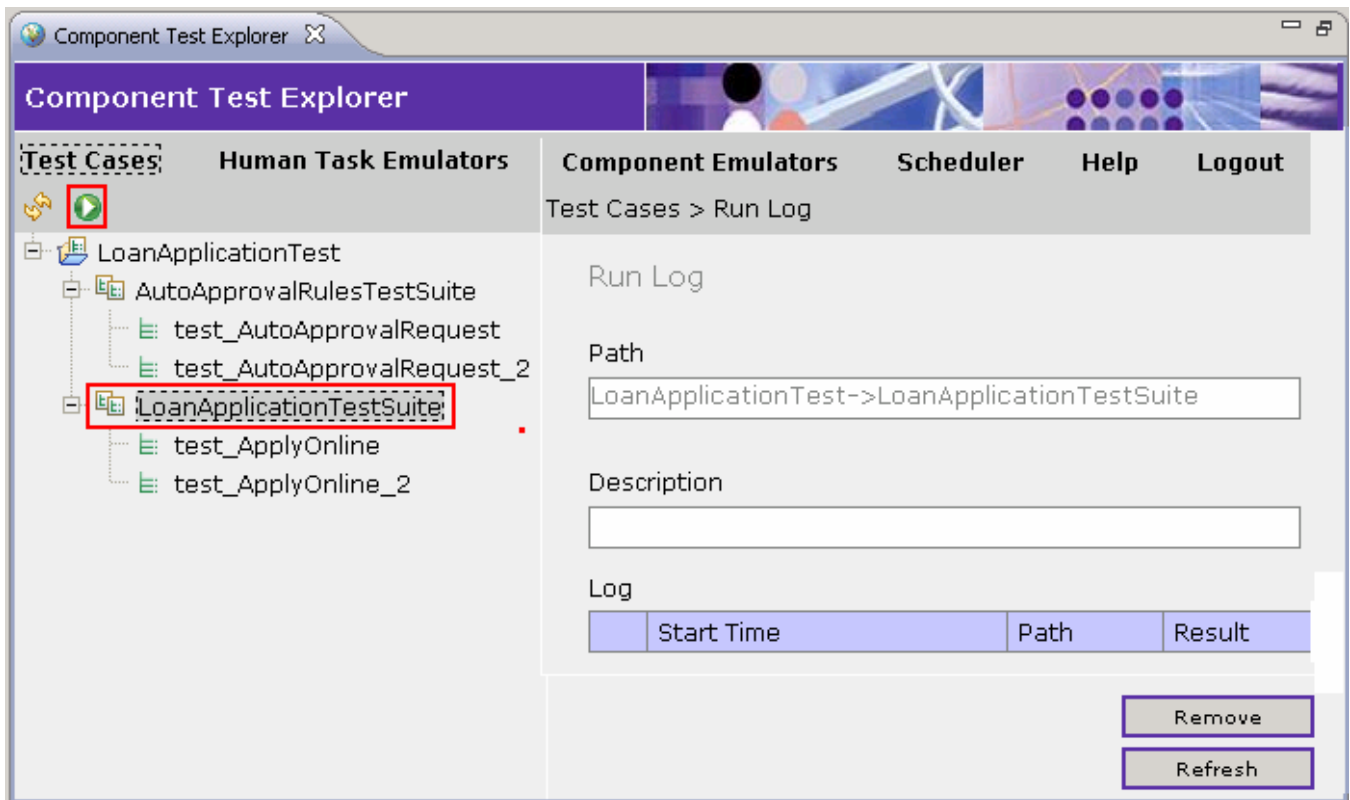
- ___ c. Click **OK** to install the **Component Test Explorer**. This only takes a few minutes.
- ___ d. When the Component Test Explorer is finished installing, you should see the following dialog



- ___ e. Click **OK**
- ___ f. Now login to the **Component Test Explorer** using the admin/admin user



__ g. In the left navigation pane, click the **LoanApplicationTestSuite**.



- ___ h. Click the **Run** button as shown in the above picture.
- ___ i. Scroll to the very right of the screen to see the test results:

Test Cases > Run Log

Run Log

Path

Description

Log

	Start Time	Path	Result	
<input type="checkbox"/>	2008-12-12 01:00	LoanApplicationTest->LoanApplicationTestSuite	pass 1 fail 1	<input type="button" value="Remove"/> <input type="button" value="Refresh"/>

- ___ j. Click the (**pass 1 fail 1**) link in the **Result** column. You can click the exception to see more info

Test Cases > Run Log > Detailed Report

Detailed Report

Start Time

End Time

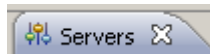
LoanApplicationTest > LoanApplicationTestSuite

Test Case Name	Result	Description
test_ApplyOnline	pass	Default pass
test_ApplyOnline_2	fail	junit.framework.AssertionFailedError: Variation: [Default] Variable:[BankResponse] FAIL(Input:[false...

- ___ k. Examine the remaining features. Click the **Logout** link and close the explorer when you are finished

___ 2. Clean up the test server.

- ___ a. Right Click the Server in the Servers View in WID.



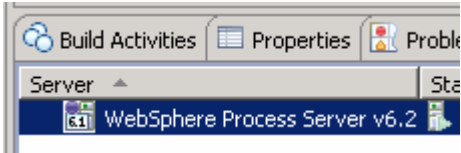
- ___ b. Select Add and Remove Projects ...
- ___ c. Click Remove All

___ d. Click Finish

___ e. Click Ok.

___ 3. Stop test server.

___ a. Select the Server



___ b. Click .

___ 4. You are done with the lab.