IBM WEBSPHERE BUSINESS IINTEGRATION V7 - LAB EXERCISE

# Integrated test client

Objectives	2
Requirements	2
Skills acquired	2
ntroduction	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	
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
  - o The ability to perform unit testing of a module or set of modules
  - o The ability to perform component testing
  - o The ability to add test cases, suites and emulations
  - o An understanding of the Component Test Explorer

### Introduction

Using WebSphere<sup>®</sup> 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.

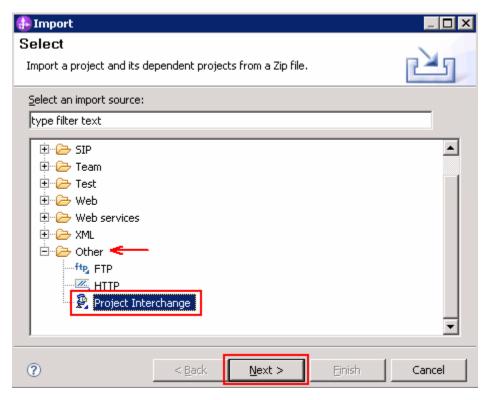
### Part 1: Importing the projects

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

🌐 Workspace Launcher		×
Select a workspace		
IBM WebSphere Integration Developer 6.2 stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.		
Workspace: C:\LabFiles70\WID\TestClient\workspace	•	Browse
Lyse this as the default and do not ask again		
OK		Cancel

- \_\_\_ 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

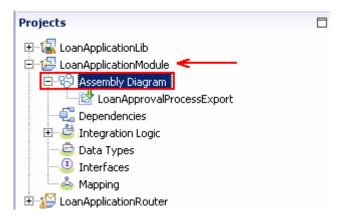
🤀 Import Project	Interchange Content	5		
Import Projects				
Import Projects from	n a zip file.			
			DT ele	
From zip file:	C:\LabFiles70\WID\Te	estClient\LoanAp	pP1.zip 💌	Browse
Project location root	C:\LabFiles70\WID\Te	estClient\workspa	ace	Browse
De LoanApplica	tionLib			
🛛 🗹 🗁 Loan Applica				
4				
Select All Desel	ect All Select Referen	ced		
0	< <u>B</u> ack	Next >	Einish	Cancel

- \_\_\_\_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:

🗄 Business Integration 🕱 🔀 Physical Resources 🗖	
t∄   ⊕   &   ⊑ \$	~
Integration Solutions	
Click to add an integration solution	
	_
Projects	
🖭 🗟 LoanApplicationLib	
🗄 🖅 LoanApplicationModule	
🗄 🕼 LoanApplicationRouter	

- \_\_\_ 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 processed 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



\_\_\_\_\_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	T 🗱 💭 🖑 🗶
Name	Rule1
Presentation	
Action	AutoApprovalResponse = false
Name	Rule2
Presentation	
IF	any of the following is true 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.

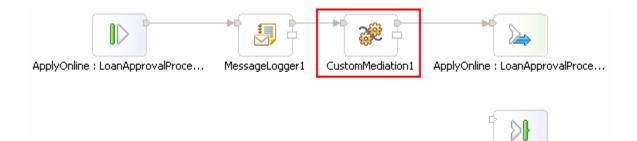
- \_\_\_\_f. .Close the rule set editor.
- \_\_\_\_ g. Close the assembly editor.
- \_\_\_\_h. View the LoanApplicationRouter mediation module. In the business integration view, expand LoanApplicationRouter and double click double click Assembly Diagram to open it in the assembly editor:

	Projects LoanApplicationLib LoanApplicationModule LoanApplicationRouter LoanApplicationRouter LoanApprovalProcessInterfaceExport1 Dependencies LoanApprovalProcessInterfaceExport1 Dependencies Data Types Mapping	
E toanApprovalProcess	InterfaceExport1	

**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



ApplyOnline : LoanApprovalProce...

\_\_\_\_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.

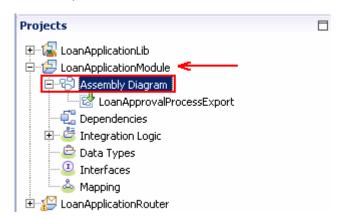
Co Build Activities Drope	
Description Terminal User Properties Details Java Imports Promotable Properties	Implementation: O Visual O Java  * GENERATED COMMENT - DO NOT MODIFY  * Variables: for output terminals - out (com.ibm.wsspi.sil  * for user properties - <no (com.ibm.wsspi.sibx.mediation.="" *="" <="" com.ibm.wsspi.sibx.mediation.mediationconfig="" de:="" exceptions:="" inputs:="" inputterminal="" properties="" th="" user=""></no>
	<pre>/* Simply print a statement indicating routing was performe System.out.println("Routing request to Loan Application"); out.fire(smo); // propagate the service message object to t </pre>

\_\_\_\_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					
å► Invoke	Configuration: Default Mod	lule Test	•			
	Module: LoanApplica	tionModule	•			
	Component: AutoApprov	alRules	•			
	Interface: AutoApprov	valRulesInterface	•			
	Operation: AutoApprov	valRequest	•			
	Initial request parameters					
	∮ 🖗 👷 📄 🗖					
	Name	Туре	Value			
	🖃 📇 AutoApprovalIng	ut LoanRequestBO	<ul> <li>✓</li> </ul>			
	🚊 🖳 🔤 ApplicantInfo	PersonalDataBO	✓			
	💭 Name	string	✓			
	💭 EmailAdo	ress string	✓			
	🛄 🛄 TaxPaye	rId int	<b>√</b> 0			
	🛄 🛄 LoanAmount		✓ 0			
			► I			

**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 TaxPayerld : 123456789 LoanAmount : 49999

- \_\_\_ d. Click the Continue button on the top left of the test client editor: (12)
- \_\_\_\_e. The **Deployment Location** dialog opens:

de Deployment Location	
Select a Deployment Location	
Specify a runtime location where this test will deploy.	
Deployment location:	
WebSphere Process Servers WebSphere Process Server v7.0 at localhost WebSphere Enterprise Service Bus Server	<u>New ≦</u> erver
Mode: Run	[
Use this location as the default and do not ask again	
① To change the default location elsewhere, open the Properties project and select Business Integration > Integration Test Clie	
?	Cancel

**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:

🚯 User Login - Default Module Test	×
Security is enabled on the selected runtime environment(s). Sign in to continue the test.	
<u>U</u> ser ID:	
admin	
Password:	
•••••	
Jse the authentication settings in the preference and <u>n</u> ever ask again	n.)
OK Cancel	

\_\_\_\_ 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	<u>M</u> odule:	LoanApplicationModule			
🏊 - 🔲 - 📄 🆆 📦 🔳 🔡	Component:	<u>AutoApprovalRules</u>			
🖃 🐩 Invoke (AutoApprovalRules:AutoApprovalRequest)	<u>I</u> nterface:	AutoApprovalRulesInte	erface.		
🖂 隆 Invoke started	Operation:	AutoApprovalRequest			
划 Invoke (AutoApprovalRules:AutoApprovalRequest)	Return param	eters:			
Return (AutoApprovalRules:AutoApprovalRequest)					
Tinvoke returned					
		Name	Туре	V	alue
		AutoApprovalResponse	boolean	🖌 true	
	•				▶
	1				

- \_\_\_\_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	Cor	mponent:	AutoApprovalRu	iles	•
<ul> <li>Invoke (AutoApprovalRules:AutoApprovalRequest)</li> <li>Invoke started</li> <li>Invoke (AutoApprovalRules:AutoApprovalRequest)</li> <li>Return (AutoApprovalRules:AutoApprovalRequest)</li> <li>Invoke returned</li> </ul>	Opt	Interface: AutoApprovalRulesInterface  Operation: AutoApprovalRequest Initial request parameters			
	ł	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Vame	Туре	Value
			oApprovalInput ApplicantInfo I Name D EmailAddress TaxPayerId LoanAmount	LoanRequestBO PersonalDataBO string	<ul> <li>✓</li> <li>✓ Joe</li> <li>✓ joe@ibm.com</li> <li>✓ 1234567</li> <li>✓ 50001</li> </ul>

\_\_\_\_ 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	Туре	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**  $\rightarrow$  **Invoke started** and select **Invoke** in the test client editor.

Events	Configuration	Default Module Tes	<u>st</u>	
🎄 - 🔲 - 📄 券 🕓 🔳 🔛	Module:	LoanApplicationMo	<u>dule</u>	
🖃 🐩 Invoke (AutoApprovalRules:AutoApprovalRequest)	Component:	<u>AutoApprovalRule</u> :	-	
🖂 🐂 Invoke started	Interface:	AutoApprovalRule:	<u>sInterface</u>	
Invoke (AutoApprovalRules:AutoApprovalRequest)	Operation:	AutoApprovalRequ	<u>iest</u>	
Return (AutoApprovalRules:AutoApprovalRequest)	Invocation par			
🖃 ĭ Invoke (AutoApprovalRules:AutoApprovalRequest)				
🖃 隆 Invoke started		Name	Туре	Value
💚 Invoke (AutoApprovalRules:AutoApprovalRequest)	1 😑 🔁 A	utoApprov. 📜	1. nn.	<ul> <li>✓</li> </ul>
🤎 Return (AutoApprovalRules:AutoApprovalRequest)		a Applican 📄 ⊆opy	Value	✓
Invoke returned		💭 Nam 💦 Add '	Value to Pool	🖌 Joe
		🖳 Ēmai 🔜 🚽	<u>-</u>	🖌 joe@ibm.com
		🎦 TaxF 🛛 Expo	<u>r</u> t to Fi <mark>le</mark>	✔ 123456789
	ĮĮ	LoanAmount	int V	✓ 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:

- \_\_\_\_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.

🖹 *AutoApprovalRules_Test 🐰	
Configurations	
Configurations	General Properties
😽 🗶 🔉 🦉 🔡	<ul> <li>Detailed Properties</li> </ul>
🗖 🧮 Test configuration Default Module Test	Modules included in this test configuration:
🖃 📥 Module LoanApplicationModule	占 LoanApplicationModule
Component and Reference Emulators LoanApprovalProcess	
😓 ManualApprover	
🌆 Human Task Emulators	
🖂 🕵 Monitors	
ස් LoanApprovalProcess.AutoApprovalRulesInterfacePartner -> A	
66 LoanApprovalProcess.ManualApproverInterfacePartner -> Mar	Login properties
ස් LoanApprovalProcessExport. <export> -&gt; LoanApprovalProces</export>	User ID: admin
🗆 違 Fine-Grained Traces	L <u>o</u> gout
🕍 LoanApprovalProcess (Process:LoanApprovalProcess)	
•	
Events Configurations	

- \_\_\_ b. Click the Add button: (1997)
- \_\_\_\_ c. From the **New Configuration dialog**, select the **Module** wizard:

🚯 New Configuration	_ 🗆 🗙
Select a Wizard Add modules to an existing test configuration.	Eo
<u>W</u> izards:	
Test Bucket Configuration Test Configuration Test Case Module Component and Reference Emulator Component and Reference Emulator Monitor Pine-Grained Trace	
⑦ < <u>Back</u> <u>Next</u> > Einish	Cancel

- \_\_\_ 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.

🚯 New Tes	t Module	
Add Mod Select the r	<b>ules</b> modules to add to this configuration.	
N <u>a</u> me:	Default Module Test	
	dules in this configuration:	Select All
	anApplicationRouter	Cl <u>e</u> ar All
1		
?	< <u>B</u> ack <u>N</u> ext >	Einish Cancel

- \_\_\_\_\_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: ( Implies the test client window) (

\_\_\_\_i. Select LoanApplicationRouter module under Detailed Properties section

$oxed{8}$ No component or import is found in the module. The test cannot co	ntinue
Events	General Properties
🎥 ▾   ☶ ▾   ⊑ ‡   🖸 🔲 目	▼ Detailed Properties
<ul> <li>Invoke (AutoApprovalRules:AutoApprovalRequest)</li> <li>Invoke started</li> <li>Invoke returned</li> <li>Invoke (AutoApprovalRules:AutoApprovalRequest)</li> <li>Invoke started</li> <li>Invoke started</li> <li>Invoke (AutoApprovalRules:AutoApprovalRequest)</li> <li>Return (AutoApprovalRules:AutoApprovalRequest)</li> <li>Invoke returned</li> <li>Invoke</li> </ul>	Configuration:       Default Module Test         Module:       LoanApplicationRouter         Component:       Interface:         Operation:       Image: Component:
	Initial request parameters       Initial request parameters       Image: Name       Name       Type

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.

Configurations	General Properties
8 × 8 8 E	✓ Detailed Properties
<ul> <li>Test configuration Default Module Test</li> <li>Module LoanApplicationModule</li> <li>Component and Reference Emulators</li> <li>LoanApprovalProcess</li> <li>ManualApprover</li> <li>Human Task Emulators</li> <li>Monitors</li> <li>LoanApprovalProcess.AutoApprovalRulesInterface</li> <li>LoanApprovalProcess.ManualApproverInterfacePa</li> <li>LoanApprovalProcessExport.</li> <li>Ene-Grained Traces</li> <li>LoanApprovalProcess (Process:LoanApprovalProces</li> </ul>	r 5
	Î.

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:

E *AutoApprovalRules_Test 🛛		
Configurations Configurations	General Properties	
88 🗶   🔉 🗞   🔡	<ul> <li>Detailed Properties</li> </ul>	
E Test configuration Default Module Test     D    Module LoanApplicationModule	Component: LoanApprovalProce	<u>:55</u>
<ul> <li>Component and Reference Emulators</li> <li>LoanApprovalProcess</li> <li>Manual Add</li> <li>Human Tas</li> <li>Monitors</li> <li>Monitors</li> <li>CoanApprovalProcess.ActoApprovalRulesInterfaceF</li> <li>LoanApprovalProcess.ManualApproverInterfacePar</li> <li>LoanApprovalProcessExport.</li> <li>LoanApprovalProcessExport.</li> <li>Fine-Grained Traces</li> <li>LoanApprovalProcess (Process:LoanApprovalProces</li> <li>Module LoanApplicationRouter</li> </ul>	Emulation O Disabled O Manual emulation O Programmatic emulation File: <not defined=""> Brgwse New</not>	
Events Configurations		

**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

🗄 *AutoApprovalRules_Test 🛛			
Events	General Properties		
↓ - □ - □ ↓ ○ ■ □	<ul> <li>Detailed Properties</li> </ul>		
<ul> <li>Invoke (AutoApprovelevelet determined of the server all equest)</li> <li>Invoke (AutoApprovelevelet determined of the server all equest)</li> <li>Invoke returne</li> <li>Invoke (AutoApprovelevelet determined of the server all equest)</li> <li>Invoke returned</li> <li>Invoke returned</li> </ul>	Detailed Properties   Configuration: Default Module T   Module: LoanApplicationM   Component: AutoApprovalRu   Interface: AutoApprovalRu   Operation: AutoApprovalRu   Operation: AutoApprovalRu   Initial request parameters   Initial request parameters	Nodule les lesInterface	✓         ✓ <t< th=""></t<>
		int	✔ 49999

#### \_\_\_\_ c. The Create Component Test Case dialog opens

🚯 Create Co	mponent Test Case	
Create Co	mponent Test Case	
Select an exi	sting test suite for the new component test case(s).	E.
Test project:		▼ <u>Ne</u> w
Test <u>s</u> uite:		▼ Ne <u>w</u>
0	Einish	Cancel

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

🤀 Create Component Test Project	_ 🗆 X
Create a Component Test Project	
Specify the name and location of the new component test project.	
Project name: LoanApplicationTest	
☑ Use <u>d</u> efault location	
Location: C:\IBM\workspace\62\TestClient2\LoanApplicationTes	Browse
Component test projects can be deployed and run on WebSphere Pro They contain component test suites and test cases for testing compo business integration modules. They may also contain integration test such as test execution traces, emulators, and data pools.	nents in
? <u>Einish</u>	Cancel

#### 2) Click Finish

🚯 Create Co	mponent Test Case		
	mponent Test Case roject does not have any test suite.		E
Test project:	LoanApplicationTest		▼ <u>Ne</u> w
Test <u>s</u> uite:			▼ <u>New</u>
?		Einish	Cancel

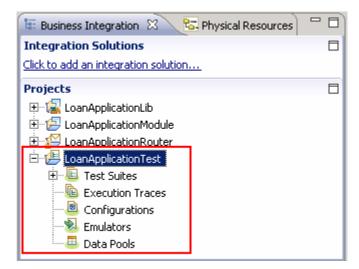
3) Now click **New** for **Test suite** to create a new Component Test Suite named, **AutoApprovalRulesTestSuite** 

🚯 Creat	e Component Test Suite	_ 🗆 🗙
Create	a Component Test Suite	L
Specify	the name and location of the new component test suite.	EE:
Project:	LoanApplicationTest	N <u>e</u> w
Folder:	TestSuite Default	Browse
Na <u>m</u> e:	AutoApprovalRulesTestSuite	
?	< Back Next > Finish	Cancel

#### 4) Click Finish

🚯 Create Co	omponent Test Case		_ 🗆 🗵
Create Co	mponent Test Case		
Select an ex	isting test suite for the new com	ponent test case(s).	E
Test project:	LoanApplicationTest		▼ <u>Ne</u> w
Test <u>s</u> uite:	AutoApprovalRulesTestSuite		▼ <u>New</u>
0		<u>F</u> inish	Cancel

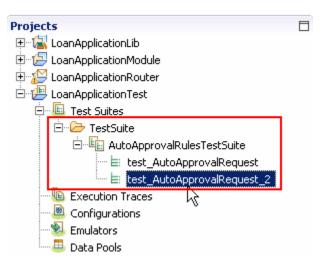
- \_\_\_ d. Click Finish
- \_\_\_\_ e. Notice the following new artifacts are created in the Business Integration view. In the Business Integration view, expand LoanApplicationTest



\_\_\_\_ 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:

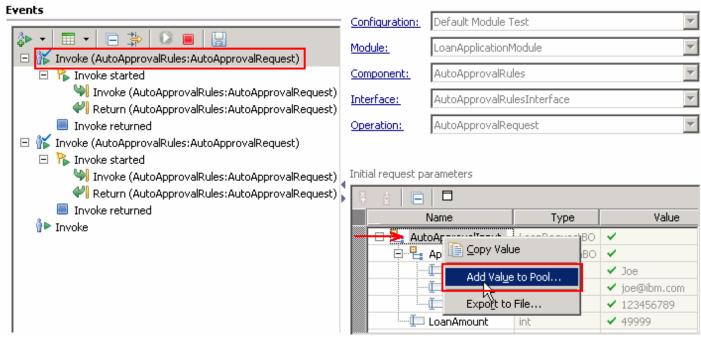
🖹 *AutoApprovalRules_Test	
Test Cases	▼ General Properties
Image: Second state       Image: Second state         Image: Second state       Image: Second state <td>Name test_AutoApprovalRequest Description</td>	Name test_AutoApprovalRequest Description
Overview Test Cases Configurations	

\_\_\_\_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

🤀 New Data Pool				
Create a new of Create a new data	•			
No data pools were A data pool stores		workspace. Nu use in the integratio	on test client. D	ata pools can
be located in a com	ponent test pr	-		
0	< <u>B</u> ack	Next >	Einish	Cancel

#### \_\_\_ c. Click Next

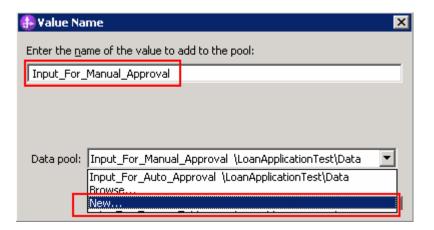
\_\_\_\_ d. In the next screen, name the data pool as Input\_For\_Auto\_Approval

🌐 New Data Pool		
Create a new d		
Create a new data	pool file in the workspace.	
Project: LoanApplic	ationTest	▼ N <u>e</u> w
Folder: Data	V D	efault Browse
Name: Input_For	_Auto_Approval	
🔽 Show only test p	rojects.	
🔽 Set this data poo	ol as the workspace default.	
(?)	< <u>B</u> ack Next >	Cancel

- \_\_\_ 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.

🌐 Value Name			×
Enter the <u>n</u> ame of the value to	add to the pool	:	
Input_For_Auto_Approval			
Data pool: Input_For_Auto_	Approval \Loan/	ApplicationTest	\Data 💌
	OK	Cancel	Show Value

- \_\_\_ 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

🥵 New Data Pool		
Create a new data pool file		
Project: LoanApplicationTest	•	N <u>e</u> w
Folder: Data	Default	Br <u>o</u> wse
Name: Input_For_Manual_Approval		
Show only test projects.		
E Set this data pool as the workspace default.		
0	Einish	Cancel

\_\_\_\_j. Click **Finish**. Ensure the new Data pool is selected

🥵 Value Na	ame	X
Enter the <u>n</u> a	ame of the value to add to the pool:	
Input_For	_Manual_Approval	
Data pool:	Input_For_Manual_Approval \LoanApplicationTest\Data	J
	OK Cancel Show Val	ue

\_\_\_ k. Click **OK** 

\_\_\_\_I. 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

Projects
🕀 🐨 🛃 LoanApplicationLib
🗄 🖅 LoanApplicationModule
🗄 🎦 LoanApplicationRouter
🖻 🕮 LoanApplicationTest
🖻 📲 Test Suites
🖻 🗁 TestSuite
E
🖹 test_AutoApprovalRequest
🔤 🗄 test_AutoApprovalReguest_2
Execution Traces
Ė
AutoApprovalRules_Test
Configurations
Emulators
🗄 🕮 Data Pools
🖃 🗁 Data
Input For Auto Approval
Input_For_Manual_Approval

# 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

Projects					
🕀 🐨 🛃 LoanApplica 🕀 🐨 🔁 LoanApplica	tionModule				
🕀 😥 LoanApplica					
🚊 🖉 Eogramppied					
	Ne <u>w</u> —	<b>&gt;</b> )	Project	•	
	Op <u>e</u> n		Component Test	Case	
· · · ·	⊆ору		🙀 Component Test S	5uite	
Executi B Configu	Paste				
- Sinnige Emulato	💢 <u>D</u> elete				
🗄 📇 Data Po	Refactor	•	_		
	🚵 Import				

\_\_\_\_b. In the Create Component Test Suite enter LoanApplicationTestSuite for Name

🤀 Create Comp	onent Test	Suite			_ 🗆 🗙
Create a Con Specify the name	•		mponent test suite.		EE:
Project: LoanAp	plicationTest			1	▼ N <u>e</u> w
Folder: TestSui	te			🔽 🔽 Defau	llt Br <u>o</u> wse
Na <u>m</u> e: LoanAp	plicationTest	5uite			
0		< <u>B</u> ack	Next >	Einish	Cancel

- \_\_\_ 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

E *AutoApprovalRules_Test	🕒 LoanApplicationTestSuite 🛛	-
Test Cases		1
Test Cases		_
📑 <mark>- 🗶 🛃</mark> 🔂 🕂		
		1
		·
Overview Test Cases Configurat	ions	

- \_\_\_\_ 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.

🥵 New Test Case	
Select a test pattern Create tests for individual operations.	E
Available test patterns:	
O < Back Next > Einish	Cancel

IBM WebSphere Integration Developer – Test Client – Lab exercise

#### \_\_\_ d. Click Next

\_\_\_\_e. In the next screen, select the operation to be tested. Expand LoanApplicationRouter → Import1 → LoanApprovalProcessInterface. Select the check box for ApplyOnline

🚯 New Test Case		
	rations to be Tested reated for each selected operation.	E
	plicationModule plicationRouter ort1 LoanApprovalProcessInterface ApplyOnline nApplicationRouter nApprovalProcessInterfaceExport1	Select All
0	< <u>B</u> ack <u>N</u> ext > Ei	nish Cancel

- \_\_\_f. Click Finish.
- \_\_\_\_\_ 3. Configure the test suite for fine-grained trace
  - \_\_\_a. Select the **Configurations** tab

E *AutoApprovalRules_Test	🗉 LoanApplicationTestSuite 🛛		- 0
Configurations			
Configurations		General Properties	
<b>S</b>		<ul> <li>Detailed Properties</li> </ul>	
🔲 🖻 🧮 Test configuration L	panApplicationTestSuite	Modules included in this test configuration:	
E 占 Module LoanApp Component Muman Task Monitors Fine-Grained	and Reference Emulators Emulators	LoanApplicationRouter	
Overview Test Cases Configurati	ons		

\_\_\_ b. Click the Add button: (1980)

\_\_\_\_ c. In the New Configuration dialog, select Module as the Wizard

🚯 New Configuration	
Select a Wizard Add modules to an existing test configuration.	E®
Wizards: Module Component and Reference Emulator Human Task Emulator Monitor Fine-Grained Trace	
Image: Second	Cancel

- \_\_\_\_ d. Click Next. Click Next again.
- \_\_\_\_e. Select check box LoanApplicationModule

🌐 New Test Mo	dule			_ <b>D</b> ×
Add Module Select the modu	<b>s</b> Iles to add to this (	configuration.		E
Name: 	LoanApplications in this configurat			
	plicationModule) plicationRouter			<u>S</u> elect All Cl <u>e</u> ar All
0	< <u>B</u> ack	Next >	<u> </u>	Cancel

- \_\_\_ f. Click Finish
- \_\_\_ g. Now expand, Module LoanApplicationModule, right-click Fine-Grained Traces. Select Add → Fine-Grained Trace from the menu

📭 *LoanApplicationTestSuite 🛛	- 8
Configurations	_
89 ×	
📃 🖃 Test configuration LoanApplicationTestSuite	
🖂 📥 Module LoanApplicationRouter	
Component and Reference Emulators	
🌆 Human Task Emulators	
er Monitors	
😕 Fine-Grained Traces	
> 🗠 Module LoanApplicationModule	•
Component and Reference Emulators	►
🆏 Human Task Emulators	
တို့ Monitors	
	rained Trace
Overview Test Cases Configurations	

\_\_\_h. From the New Fine-Grained Trace dialog, select LoanApplicationModule

🚯 New Fine-Gra	ined Trace	
Select a Mod To configure white	l <b>ule</b> :h components will be traced, select a mo	odule.
- 🗠 Mod	figuration LoanApplicationTestSuite ule LoanApplicationModule ule LoanApplicationRouter	
?	< Back Next >	Einish Cancel

\_\_\_ i. Click Next

\_\_\_\_j. In the next screen, select the LoanApprovalProcess check box

🚯 New Fine-Grain	ed Trace				
Select Compo	Select Components to Trace				
Select the process, state machine, and mediation flow components to trace in this module.					
Module n <u>a</u> me:	LoanApplicatio	onModule			
Choose the compor	ents:				
LoanAppro	valProcess			<u>S</u> elect All	
				Cl <u>e</u> ar All	
1					
?	< <u>B</u> ack	<u>N</u> ext >	Einish	Cancel	

- \_\_\_ k. Click Finish.
- \_\_ I. 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

E *LoanApplicationTestSuite	
Configurations         Image: Second system         Image: Second system      <	<ul> <li>▶ General Properties</li> <li>▶ Detailed Properties</li> <li>Available Variables by Scope</li> <li>♥ ● Global Variables</li> <li>● ApplicationInput</li> <li>● BankResponse</li> <li>● AutoApprovalData</li> <li>● AutoApprovalResponseData</li> <li>● ManualApprovalData</li> </ul>
E      Fine-Grained Traces     E      LoanApprovalProcess (Process:LoanApprov     Overview Test Cases Configurations	

\_\_\_\_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

E *LoanApplicationTestSuite	
Configurations	
89 ×	
🖃 📃 Test configuration LoanApplicationTestSuite	
🖂 📥 Module LoanApplicationRouter	
Component and Reference Emulators	
😽 Human Task Emulators	
See Monitors	
😕 Fine-Grained Traces	
->- 🗠 Module LoanApplicationModule 🛛 🕢	
Component and Reference Emulators	
🚽 🗢 🧖 Human Task Emulators	
🖗 Monitors 🛛 📥 🏧 Human Task Emulator	
🗆 🗁 Fine-Grained Traces 🛛 🗱 Bemove 🛛 🖄	
🗠 LoanApprovalProcess (Process:LoanApprov	
Overview Test Cases Configurations	

\_\_\_\_b. In the New Human Task Emulator, ensure Module LoanApplicationModule is selected.

🥵 New Huma	in Task Emulator			
Select a M Select a mode	lodule ule where new emulators (	will be created.		
	le to continue:			
	configuration LoanApplica Module LoanApplicationMo Module LoanApplicationRo	dule		
0	< <u>B</u> ack	<u>N</u> ext >	Einish	Cancel

\_\_\_ c. Click Next

\_\_\_\_d. In the next screen, select **Standalone** radio button and then select the check box for **ManualApprover** 

🚯 New Human Tas	k Emulator		
Add Human Ta Select the inline and module.		ors man tasks to emulate in this	
Module n <u>a</u> me:	LoanApplica	ationModule	
Emulate C <u>I</u> nline		• Standal <u>o</u> ne	
Emulated human tas		e:	Select All
Filter		Any)	
0	< <u>B</u> ack	Next > Einis	h Cancel

\_\_\_\_e. Click Finish. ManualApprover appears under Human Task Emulators:

🖂 📥 Module LoanApplicationModule
💭 Component and Reference Emulators
🖃 🆏 Human Task Emulators
🧖 ManualApprover
60 60 Monitors
🗆 違 Fine-Grained Traces
😫 LoanApprovalProcess (Process:LoanApprov

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

Configurations	General Properties
89 X	<ul> <li>Detailed Properties</li> </ul>
<ul> <li>Test configuration LoanApplicationTestSuite</li> <li>Module LoanApplicationRouter</li> <li>Component and Reference Emulators</li> <li>Human Task Emulators</li> <li>Monitors</li> <li>Fine-Grained Traces</li> <li>Module LoanApplicationModule</li> <li>Component and Reference Emulators</li> <li>Human Task Emulators</li> <li>Monitors</li> <li>Fine-Grained Traces</li> <li>Monitors</li> <li>Fine-Grained Traces</li> <li>Component Process (Process:LoanApprover)</li> </ul>	Iask:       ManualApprover         Component:       ManualApprover         Emulation       Image: Component of the second s

- \_\_\_\_g. Click New. The New Emulator dialog is launched
- \_\_\_\_h. Name the emulation as ManualApproverEmulator

🚯 New Emulator
Emulator
Create a programmatic emulator that emulates an interface, component or reference in a module.
Project: LoanApplicationTest
Folder: Emulator Default Brgwse
Name: ManualApproverEmulator
Show only test projects.
< Back         Mext >         Finish         Cancel

\_\_\_i. Click Next

\_\_\_\_j. In the next screen, select the ManualApprovalInterface to emulate

🚯 New Emulator	_ 🗆 🗙
Select an Interface	
Select an interface that will be emulated programmatically.	
Emulate	
© Interface	
C ⊆omponent or reference	
Select an interface to emulate:	
🖂 🖴 LoanApplicationLib	
ManualApproverInterface - http://LoanApplicationModule/ManualApprove	erInterface
< Back	Cancel

#### \_\_\_ k. Click Next

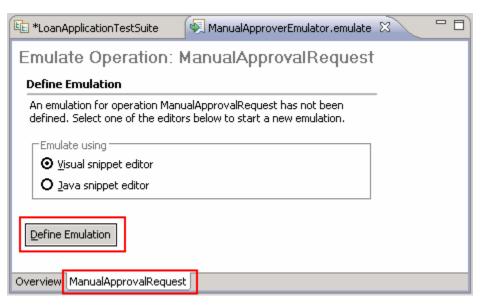
.

\_\_\_ I. 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:

Projects	
🗄 🐨 🔄 LoanApplicationLib	
🗄 🖅 LoanApplicationModule	
🗄 🎦 LoanApplicationRouter	
🖻 🕮 LoanApplicationTest	
🖻 – 💷 Test Suites	
🖻 🗁 TestSuite	
🖻 🖳 🖳 AutoApprovalRulesTestSuite	
🔤 🗄 test_AutoApprovalRequest	
test_AutoApprovalRequest_2	
E LoanApplicationTestSuite	
Execution Traces	
E C-Run	
AutoApprovalRules_Test	
Configurations	
Emulators	
ManualApproverEmulator	
🖻 – 📇 Data Poois	
🖻 🗁 Data	
Input_For_Auto_Approval	
Input_For_Manual_Approval	

IBM WebSphere Integration Developer – Test Client – Lab exercise

\_\_\_\_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: ( Return ) and then drop it on the canvas by click the white space

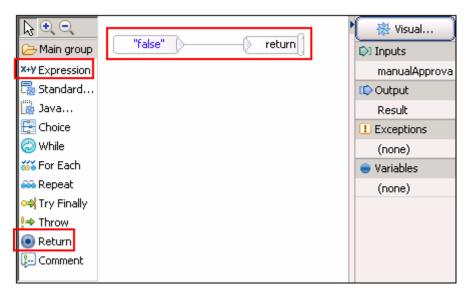


\_\_\_ p. Now, click the Expression button from the palette: ( X+Y Expression ) and then drop it over the canvas by clicking on the white space



\_\_\_\_q. Enter a value of "false". Include the quotation marks because it is a string.

<u>(</u>	return
	"false"



\_\_\_\_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.

🗄 LoanApplicationTestSuite 🕱 🛛 🗖 🗖				
Test Cases	<ul> <li>General Information</li> </ul>			
Image: Second state state       Image: Second	Description Invoke: ApplyOnline()	<b></b>		
	<ul> <li>Detailed Properties</li> </ul>	<u>_</u>		
	Module: LoanApplicationRouter	•		
	Component: Import1	•		
	Interface: LoanApprovalProcessInterface	•		
	Operation: ApplyOnline	•		
	Asynchronous mode			
	C Request C Response Exception			
	ApplicationInput: LoanRequestBO ApplicationInput	•		
Overview Test Cases Configurations				

\_\_\_\_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

ି	) Build Activities 🔲 Properties 🔝 Pr	oblems 🖗 Servers 🕃 Migration Re	sults 📧 Test Data Table 🛛	- 8		
Im	Imports Default					
I T	oanApplicationTestSuite : test_Aj	oplyOnline				
	(Dame)	🍋 Туре	[⊃]In	Expected		
	Invoke Import1:ApplyOnline(A					
	🖻 📴 ApplicationInput	LoanRequestBO	×			
		PersonalDataBO	*			
	🛄 LoanAmount	int	✓ 0			
	🛄 BankResponse	string		*		

\_\_\_\_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

Emulators
🖻 🗁 Data
Input_For_Auto_Approval
🛄 Input_For_Manual_Approval

\_\_\_\_f. In the Data Pool editor, right click Input\_For\_Auto\_Approval and select Copy Value

LoanApplicationTestSuite	🔲 Input_For_Auto_Approval	🛛 🔲 Input_For_Ma	anual_Approval 🗋 🗖 🗖
Eilter for type:			
↑ ↓ ¥ ± □			
Name	Туре	Value	
Provide the second seco	Copy <u>R</u> ow	✓ ✓ Joe ✓ joe@ibm.com ✓ 123456789 ✓ 49999	
	Import from File	_	
Input_For_Auto_Approval	Expo <u>r</u> t to File		

- \_\_\_\_ g. Close the Data Pool editor.
- \_\_\_h. Now, in the **Test Data Table** editor, right-click **ApplicationInput** and select **Paste Value**:

🗟 Build Activities 🔲 Properties 🔝 Problems 🔇	🗱 Servers 🕃 Migration Results 📧 T	est Data Table 🛛	₽~▽□₽			
Imports Default	Imports Default					
LoanApplicationTestSuite : test_ApplyOnlin	ne					
(Name	🍆 Туре	[⊳]In	Expected			
	-panRequestBO	¥				
Copy Row	ersonalDataBO	8				
Loan Paste Row Before	lt	✓ 0				
Verify Ap 💼 Paste Row After						
BankResi Select <u>A</u> ll	ring		26,			
Copy Value						
💼 <u>P</u> aste Value						
Copy <u>V</u> ariable Name						

\_\_\_\_i. Enter "Your application has been approved." for the **BankResponse** 

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

Build Activities 🔲 Properties	s 🔝 Problems [	🔋 Server Logs 🗍	Servers 🖪 *Test Data Table 🛛	- 0	
Imports Default					
LoanApplicationTestSuite :	test_ApplyOnlin	e			
Name	🍫 Туре	闷 In	Expected		
🚊 🖳 ApplicationInput	LoanRequestBO	×			
🖻 🖳 ApplicantInfo	PersonalDataBO	×			
🏳 Name	string	🗸 Joe			
EmailAddress	string	🗸 joe@ibm.com			
🛄 🛄 TaxPayerId	int	<ul><li>123456789</li></ul>		1	
🛄 LoanAmount	int	✓ 49999			
Verify ApplyOnline				7	
🛄 💭 BankResponse	string		⊱ Your application has been approved. 4	1	

\_\_\_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.

🔃 *LoanApplicationTestSuite 🛛 🕄	Input_For_Auto_Approva	🔲 📰 Input_For_Manual_Approv	val 🗋 🗖 🗖
- F • 🗙 🔙 🗘 🕂 🗐	E	Interface: LoanApprovalPr	ocessInterface
🖃 🗄 test_ApplyOnline		Operation: ApplyOnline	<b>•</b>
🔷 🔹 🕈 Invoke Import1:Ap	pplyOnline	, 🗖 Asynchronous mode	
🗆 🗄 test_ApplyOnline_2			
🚽 Invoke Import1:Ap		·	
👗 Wait for LoanAppl	icationBackendSLA (ms)	Request 3 Response	Exception
		ApplicationInput: LoanReques	tBO ApplicationInput
Overview Test Cases Configuration	าร		
😚 Build Activities 🖾 Properties [	🖞 Problems 🛛 🛅 Server Logs 🛛	🔁 Servers 🔣 🔣 *Test Data Table 🛽 🕹	3 - 8
Build Activities 🔲 Properties [		🔀 Servers 🔚 *Test Data Table 💈	3
· · · · · · · · · · · · · · · · · · ·		移 Servers 📴 <u>*Test Data Table 8</u>	3 C C
LoanApplicationTestSuite : tes	t_ApplyOnline_2		
LoanApplicationTestSuite : tes	t_ApplyOnline_2	 []]In	
LoanApplicationTestSuite : tes	t_ApplyOnline_2	 ✓	
LoanApplicationTestSuite : tes	t_ApplyOnline_2 Type LoanRequestBO PersonalDataBO	 ✓ ✓	
LoanApplicationTestSuite : tes	t_ApplyOnline_2 Type LoanRequestBO PersonalDataBO string		
LoanApplicationTestSuite : tes	t_ApplyOnline_2 Type LoanRequestBO PersonalDataBO string string	✓     ✓     ✓     ✓     ✓     ✓     ✓     ✓     ✓     ✓     ✓     ✓     ✓     joe@ibm.com	
LoanApplicationTestSuite : tes	t_ApplyOnline_2 Type LoanRequestBO PersonalDataBO string string int		Expected
LoanApplicationTestSuite : tes	t_ApplyOnline_2 Type LoanRequestBO PersonalDataBO string string int		

\_\_\_I. 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.

Add 🔸	📑 Add Test Case
💢 Remove	🗳 Add Invocation
kun Test	Add Wait On
	\overline Add Wait For Time 🔪
Refresh	

\_\_\_\_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...

Test Cases	<ul> <li>General Information</li> </ul>
Image: Second system       Image: Second system       Image: Second system       Image: Second system         Image: Second system       Image: Second system       Image: Second system       Image: Second system         Image: Second system       Image: Second system       Image: Second system       Image: Second system       Image: Second system         Image: Second system       Image: Second	Description
Invoke Import 1: ApplyOnline	<ul> <li>Detailed Properties</li> </ul>
Wait for 50 (ms)	Amount of time to wait (ms): 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.

🚯 New Data Table	¥ariable	×
Name: LoanApplicat	tionBackendSLA	
Type: long		Br <u>o</u> wse
Add: Before 💌	Invoke Import1:ApplyOnline(App	olicationInput)
Type system Java <u>X</u> SD	Invoke Import 1 : ApplyOnline(App ApplicationInput Verify ApplyOnline BankResponse	olicationInput)
Variable intent • Input paramete • Expected result		
Array Array Dimension: 1		
	Ок	Cancel

#### 3) Click OK

4) The Test Data Table view now displays the wait before the invoke of import1

I

Name	🍆 Туре	[≫]In	Expected
💼 💭 LoanApplicationBackend:	5L long	✓ 0	
	hlir		
🚊 🖳 ApplicationInput	LoanRequestBO	×	
🚊 🖳 ApplicantInfo	PersonalDataBO	×	
💭 Name	string	🗸 Joe	
🖳 🛄 EmailAddress	string	🗸 joe@ibm.com	
🛄 TaxPayerId	int	✓ 1234567	
🛄 LoanAmount	int	✓ 50001	
📖 🗊 BankResponse	string		( <del>*=</del> )

### LoanApplicationTestSuite : test\_ApplyOnline\_2



LoanApplicationTestSuite : test_ApplyOnline_2					
	💭 Name	🤏 Туре	[≫]In	Expected	
	💬 💭 LoanApplicationBackendSL	long	✓ 10000		

\_\_\_\_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

Projects	🛃 Export
🕀 🐨 🛃 LoanApplicationLib 🕀 🐨 🔁 LoanApplicationModule	Refresh Add to Asset Repository
Englishing CoanApplicationRouter Englishing LoanApplicationTest Englishing Test Suites	<u>帰</u> Run Test Validate
E-E TestSuite	Team  Compare With Replace With Properties
<ul> <li>IcoanApplicationTestS</li> <li>Itest_ApplyOnline</li> <li>Itest_ApplyOnl</li></ul>	_2 roval

- \_\_\_\_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			
₽ - =	<ul> <li>Detailed Properties</li> </ul>			
<ul> <li>■ Invoke Import1:ApplyOnline</li> </ul>	Configuration: LoanApplicationTestSuite   Verdict : Failed   Total: 2/2   Passed: 1   Failed: 1   Failed: 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
\$•• ■• = \$• 0 ■	<ul> <li>Detailed Properties</li> </ul>
<ul> <li>□</li> <li>□</li></ul>	Module:       LoanApplicationRouter         Component:       Import1         Interface:       LoanApprovalProcessInterface         Operation:       ApplyOnline         Return parameters:
Return (Import1:ApplyOnline) Test Case (test_ApplyOnline_2) [Failed]	Image: State of the state o
	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.

Events	General Properties	
▶ - ■ - ⊝ ╠ ◎ ■ 🔒	<ul> <li>Detailed Properties</li> </ul>	
<ul> <li>Run Test (LoanApplicationTestSuite) [Failed]</li> <li>Test Suite (LoanApplicationTestSuite) [Failed]</li> <li>Test Case (test_ApplyOnline) [Passed]</li> <li>Test Variation (Default) [Passed]</li> <li>Test Variation (Default) [Passed]</li> <li>Test Case (test_ApplyOnline</li> <li>Test Case (test_ApplyOnline_2) [Failed]</li> </ul>	Test variation: Default Verdict : Failed Results:	-
Test Variation (Default) [Failed]	Name Ty Actual Expected	_
🕀 🜩 Invoke Import1:ApplyOnline	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 in not installed to the test server. If this is the first time you launched the Component Test Explorer, you will be prompted for installation

🚯 Com	Component Test Explorer Installation						
It has been detected that the Component Test Explorer is not installed on the server. Would you like the Component Test Explorer application to be automatically installed on the server for you?							
	<u>Y</u> es <u>N</u> o						

- \_\_\_\_ 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

🚯 Com	🔒 Component Test Explorer Installation 🛛 🔀					
٩	The Component Test Explorer application has been successfully installed on the server. The application name is ComponentTestExplorer. If needed, this application can be uninstalled using the Administrative Console of the server. Note that the Component Test Explorer application can be accessed from other Web browsers by using the URL: http:// <hostname>[:<port>]/ComponentTestExplorer.</port></hostname>					
	(OK					

- \_\_\_ e. Click OK
- \_\_\_\_f. Now login to the **Component Test Explorer** using the admin/admin user

📀 Componer	it Test Explorer 🛛 🖓 🖓							
Component Test Explorer								
	Log in							
	If WebSphere Process Server security is not enabled, you can use any user name or password here. Otherwise use the WebSphere Process Server administrator user name and password. <b>User Name</b> admin							
	Password •••••							
	LOGIN							

# \_\_\_ g. In the left navigation pane, click the LoanApplicationTestSuite.

Somponent Test Explorer				- e
Component Test Explorer		X		
Test Cases Human Task Emulators	Component Emulators Test Cases > Run Log	Scheduler	Help	Logout
E LoanApplicationTest	Run Log			
<pre>test_AutoApprovalRequest     test_AutoApprovalRequest_2     test_AutoApprovalRequest_2     test_AutoApprovalRequest_2     test_ApplyOnline </pre>	Path LoanApplicationTest->	>LoanApplicati	onTestSu	ite
└── ⊨ test_ApplyOnline_2	Description			
	Log Start Time	Pa	ath	Result
				Remove Refresh

- \_\_\_ 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:

Tes	st Ca	ises > Run Log				
	Run	i Log				
	Path	ı				
	LoanApplicationTest->LoanApplicationTestSuite					
	Description					
	Log					
		Start Time	Path	Result		Remove
		2008-12-12 01:00	LoanApplicationTest->LoanApplicationTestSuite	pass 1 fail 1		Refresh

\_\_\_\_j. Click the ( pass 1 fail 1) link in the **Result** column. You can click the exception to see more info

Tes	Test Cases > Run Log > Detailed Report							
Detailed Report								
	Start Time							
	2008-12-12 01:00:11.42	1						
	End Time							
	2008-12-12 01:00:19.906							
	LoanApplicationTest > LoanApplicatiouite							
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
- 6 October 2010

IBM WebSphere Integration Developer – Test Client – Lab exercise

### \_\_\_ d. Click Finish

\_\_\_e. Click Ok.

- \_\_\_\_ 3. Stop test server.
  - \_\_\_a. Select the Server

🗞 Build Activities 🔲 Properties 🔝 Pr	ople
Server 🔺	Sta
WebSphere Process Server v6.2	₽.

\_\_ b. Click 💻

\_\_\_\_\_4. You are done with the lab.