#### IBM WEBSPHERE PROCESS INTEGRATION 6.0 – LAB EXERCISE

## **CleansePublish BPEL**

What this exercise is about
Lab Requirements1
What you should be able to do1
Introduction2
Exercise Instructions
Part 1: Initialize the Workspace for this Lab Exercise
Part 2: Construct CleansePublish BPEL Business Process
Part 3: Assemble CleansePublish15
Part 4: Unit Test CleansePublish Business Process
What you did in this exercise
Solution Instructions
Task: Adding Remote Server to WebSphere Integration Developer Test Environment

## What this exercise is about

Business Process Execution Language (BPEL) is a standard-based language which can be used to describe the behavior of business processes which are composed of services. Combining BPEL with the SCA programming model allows for the coordination of individual and independent SCA services into much larger units of work and business transactions. Individual SCA services can be brought together and also benefit from the advanced capabilities of event handling, fault handling, and compensation. In this lab, you will create a simple BPEL business process using WebSphere Integration Developer. The business process will use interfaces which are part of the SCA programming model to complete a simple sequence of steps. After assembling the BPEL business process as a service (SCA component), you will component test the business process.

## Lab Requirements

List of system and software and other tasks required for the student to complete the lab.

- WebSphere Integration Developer V6.0.1 installed
- WebSphere Process Server V6.0 test environment installed
- Sample code in the directory C:\Labfiles60 (Windows) or /tmp/LabFiles60 (Linux)

## What you should be able to do

At the end of this lab you should be able to:

- Construct a simple BPEL business process
- Assemble a BPEL business process following the SCA programming model
- Component test a BPEL business process with the Integration Test Client

## Introduction

The following diagram highlights the part of the overall scenario that will be addressed in this lab. You create a WSBPEL business process which will be responsible for coordinating the normalization of the data coming from the PeopleSoft module and passing onto the Siebel and Flat File adapters. The data will be normalized using SCA components. As part of the business process design, you will use the interfaces of the SCA components when defining the sequence of steps. The first and second steps (AutoClean and ManualClean) will be an invocation of the Clean interface. The first can be wired to a Business Rule, Selector, or Java<sup>™</sup> implementation to provide an automated cleansing of the data. The second can be wired to a human task to allow for manual cleansing of the data. Because SCA components are used, at any time the implementation can be changed to any type which matches the clean interface. The third step in the business process is a one way request to publish the normalized data to the Siebel and Flat File modules.



## **Description of the Module**

The following diagram shows the ManualClean module. It highlights the parts of the module that will be addressed in this exercise.



## **Business Objects**

The following generic business objects (GBOs) will be used in this exercise.

- Clip The parent business object describing characteristics of a particular clip. This generic business object originates from the application specific business object (ASBO) from the PeopleSoft system. In the complete solution, a map and mediation is used to transform the application specific business object from the PeopleSoft system to the generic business object which the Manual Clean human task can operate upon.
- ClipItem The child business object describing a retail packing option for this type of clip. This generic business object is also defined from the application specific business object (ASBO) from the PeopleSoft system and transformed to the generic business object using a map and mediation.

In addition to these business objects, there will be a business graph generated to contain these business objects.

ClipBG

Although this business graph is generated by the tools, it is only associated with the parent business object. The business graph for the child business object is not used in the scenario.

#### Interfaces

The following interface is generated by WebSphere Integration Developer and uses the business graph and business objects.

**CleansePublish** – Contains a single one-way operation that passes a single part named inClipBG of type ClipBG on the request. This interface will be used on the definition of the business process and exposed as part of the exported SCA binding which is called by other SCA components.

**Clean** – Contains a single Request/Response operation that passes a single part named inClipBG of type ClipBG on the request and a single part on the response, named outClipBG of type ClipBG. With the use of this interface, it is expected that any changes made to the ClipBG will be passed directly back. This interface will be used to define two Reference Partners in the business process. One will be for the AutoClean invoke step and the other for the ManualClean invoke step. The use of the same interface simplifies the data exchange within the business process. The response from AutoClean can be passed directly to the request on ManualClean as they are the same message type.

**Publish** - Contains a single one-way operation that passes a single part named inClipBG of type ClipBG on the request. This interface will also be used to define a Reference Partner in the business process and will be used on the Publish invoke step. With the interface using the same message type as the Clean interface, the response from ManualClean can be directly passed to the request on Publish.

## **Exercise Instructions**

Some instructions in this lab might be specific for Windows platforms. If you run the lab on a platform other than Windows, you will need to run the appropriate commands, and use appropriate files (for example .sh in place of .bat) for your operating system. The directory locations are specified in the lab instructions using symbolic references as follows:

Reference Variable	Windows Location	Linux Location
<wid_home></wid_home>	C:\Program Files\IBM\WebSphere\ID\6.0	/opt/IBM/WebSphere/ID/6.0
<wps_home></wps_home>	<wid_home>\runtimes\bi_v6</wid_home>	<wid_home>/runtimes/bi_v6</wid_home>
<lab_files></lab_files>	C:\Labfiles60	/tmp/Labfiles60
<workspace></workspace>	C:\Labfiles60\eXchange\CleanPublishBPEL\ workspace	/tmp/Labfiles60/eXchange/ CleanPublishBPEL/workspace
<temp></temp>	C:\temp	/tmp
<solution></solution>	C:\Labfiles60\CleansePublishBPEL\Solution	/tmp/Labfiles60/CleansePublishBPEL/ Solution

**Windows users' note**: When directory locations are passed as parameters to a Java program such as wsadmin, you must replace the backslashes with forward slashes to follow the Java convention. For example, C:\LabFiles60\ would be replaced by C:/LabFiles60/.

Note that the previous table is relative to where you are running WebSphere Integration Developer. The following table is related to where you are running the remote test environment:

Reference Variable	Example: Remote Windows test server location	Example: Remote z/OS <sup>®</sup> test server location	Input your values for the remote location of the test server
<server_name></server_name>	server1	cl1sr01	
<was_home></was_home>	C:\Program Files\IBM\WebSphere\AppServer	/etc/cl1cell/AppServerNode1	
<hostname></hostname>	localhost	mvsxxx.rtp.raleigh.ibm.com	
<bootstrap_port></bootstrap_port>	2809	2809	
<telnet_port></telnet_port>	N/A	1023	
<profile_name></profile_name>	AppSrv01	default	
<userid></userid>	N/A	cl1admin	
<password></password>	N/A	fr1day	

Instructions for using a remote testing environment, such as z/OS, AIX<sup>®</sup> or Solaris, can be found at the end of this document, in the section "<u>Task: Adding Remote Server to WebSphere Integration Developer</u> <u>Test Environment</u>".

#### Part 1: Initialize the Workspace for this Lab Exercise

\_\_\_\_1. Follow the directions below to initialize the Workspace using the following values values:

#### <WORKSPACE>

C:\Labfiles60\eXchange\CleanPublishBPEL\workspace

#### <PROJECT\_INTERCHANGE>

C:\Labfiles60\eXchange\CleansePublishLibrary\import\CleansePublishLibrary\_PI.zip

#### <MODULE>

CleansePublishBPEL

#### <DEPENDENT\_LIBRARIES>

CleansePublishLibrary

- 2. Start WebSphere Integration Developer V6.0.1 with a new workspace located at < WORKSPACE>.
  - \_\_\_\_a. From Windows Explorer, navigate to the <WID\_HOME> directory and double click on wid.exe
  - \_\_\_\_b. When prompted for workspace name, enter the value provided by the **<WORKSPACE>** variable for this lab and click **OK**.

Workspace Launcher	×
<b>Select a workspace</b> IBM WebSphere Integration Developer stores your projects in a directory called Select the workspace directory to use for this session.	a workspace.
Workspace: C:\Labfiles60\eXchange\CleanPublishBPEL\workspace	Browse
Use this as the default and do not ask again	OK Cancel

\_\_\_\_ c. When WebSphere Integration Developer V6.0 opens, close the **Welcome page** by clicking on the Go to the workbench icon (bent over arrow at top-right).



- \_\_\_\_d. Ensure you are in the **Business Integration** perspective.
- \_\_3. If this lab requires you to import a project interchange file, setup the required libraries and modules for this lab by importing the project interchange file <PROJECT\_INTERCHANGE>.
  - \_\_\_\_a. From the menu bar, select File -> Import...
  - \_\_\_\_b. In the Import dialog, scroll down and select Project Interchange.

🚯 Import			×
Select Import a project and its d	ependent projects from a Z	ίp file.	Ľ
Select an import source:			
<ul> <li>Log File</li> <li>Performance Call Grap</li> <li>Probe</li> <li>Profiling file</li> <li>Profiling filter</li> <li>Project Interchange</li> <li>RAR file</li> </ul>	h		
	< <u>B</u> ack <u>N</u> ext :	> Einish	Cancel

- \_\_\_ c. Click Next.
- \_\_\_\_d. In the Import Projects dialog, initialize the From zip file: field to <**PROJECT\_INTERCHANGE>.**
- \_\_\_\_e. Click the Select All button.

🚯 Import Project Interchange Contents	×
Import Projects	
Import Projects from a zip file.	
	18
From zip file: C:\Labfiles60\eXchange\CleansePublishLibrary\impor	Browse
Project location root: C:\Labfiles60\eXchange\CleanPublishBPEL\workspace	Browse
CleansePublishLibrary	
Select All Deselect All Select Referenced	
< <u>B</u> ack <u>N</u> ext > <u>Finish</u>	Cancel

- \_\_\_f. Click Finish.
- 4. If this lab requires that you create a Business Integration module called **<MODULE>**, complete these steps:
  - \_\_\_\_a. Right click on the background of the Business Integration view to access the pop-up menu.
  - \_\_\_\_b. Select New > Module.

🗄 Business Integration 🗙	
	# ▷ 🖻 🕏 🔻
🕀 🔁 CleansePublishLibrary	
Mau	
Ne <u>w</u>	Project
Show Files	🖄 Mediation Module
The second	📂 Module
Copy	😭 Library

\_\_\_\_ c. In the New Module dialog, enter **<MODULE>** for the Module Name.

🔂 New Module
Module Create a new business integration module. A module is a project that is used for development, version management, organizing resources, and deploying to the runtime environment.
Module Name       CleansePublishBPEL         Module Location       Image: Clean PublishBPEL         Image: Clean PublishBPEL       Browse         Directory:       C:\Labfiles60\eXchange\Clean PublishBPEL\workspace\Clean sePublishBPEL         Business integration modules can be deployed and run on WebSphere Process Server. They can contain many types of components, such as business processes, assembled together for the purpose of business integration.
< <u>Back</u> <u>N</u> ext > <u>Finish</u> Cancel

- \_\_\_\_d. Click Finish.
- 5. If this lab requires that **<MODULE>** needs any **<DEPENDENT\_LIBRARIES>**, complete these steps.
  - \_\_\_\_a. In the Business Integration view, right click on the **<MODULE>** you just created to access the pop-up menu.
  - \_\_\_\_b. Select Open Dependency Editor.



- \_\_\_\_ c. In the Dependency Editor, click the **Add...** button.
- \_\_\_\_d. In the Library Selection dialog, select from the list the <DEPENDENT\_LIBRARIES>.
- \_\_\_ e. Click **OK.**
- \_\_\_\_f. Save (Ctrl+S) and close the dependency editor.

### Part 2: Construct CleansePublish BPEL Business Process

In this part, you will create a business process which will combine the execution of separate components into a single automated unit.

- 1. Create an empty business process named CleansePublishBPEL.
  - \_\_\_\_a. Expand CleansePublishBPEL and Business Logic in the Business Integration view.
  - \_\_\_\_b. Right click on Processes and select **New > Business Process**.
  - \_\_\_\_ c. Type in **com.clipsandtacks.process** for the Folder.
  - \_\_\_\_d. Enter CleansePublishProcess for the Name.

🚝 New Busines	s Process		×
New Business F	Process		
<u>M</u> odule:	CleansePublishBPEL	New	
Namespace:	http://CleansePublishBPEL/com/clipsandtacks/proc	🔽 Default	
F <u>o</u> lder:	com.clipsandtacks.process	Browse	
N <u>a</u> me:	CleansePublishProcess		

- \_\_\_e. Click Next.
- \_\_\_\_\_f. Select the radio button for Select an Existing Interface.
- \_\_\_\_g. Click the **Browse...** button and select **CleansePublish** from the list. Click **OK**.
- \_\_\_\_h. You will see the operation "cleanseAndPublishClip" appear. Click Finish. The business process will be created and opened in the editor.
- \_\_\_2. The CleansePublishProcess is a simple business process composed of a series of steps that form the business logic. Define the business logic with Invoke activities.
  - \_\_\_\_a. Select the **Invoke** activity from the activity palette on the left side of the editor.

👤 Cleans	ePublishProcess 🗙
R	
Ξ.	
0	
V	
~ *	
•	

- \_\_\_\_ b. Drop 3 Invoke activities under the **Receive** activity. The Invoke activities carry out requests to different service components to execute different parts of the business process.
- \_\_\_\_\_c. Change the name of the activities to **AutoClean**, **ManualClean**, and **Publish** by selecting the activity and changing it directly on the icon or select the activity and then the **Properties** view, and select the **Description** tab.



- 3. The various steps in the business process need services to call. These services are defined by interfaces and called partner links. Create the partner links for the process.
  - \_\_\_\_a. In the Business Integration view, expand CleansePublishLibrary and Interfaces.
  - \_\_\_\_b. Drag and drop **Clean** onto the process editor. You should see the Interface appear on the left side under the **Reference Partners** tab.
  - \_\_\_\_ c. Drag and drop **Clean** onto the process editor again.
  - \_\_\_\_d. Drag and drop **Publish** onto the process editor.
  - \_\_\_\_e. Select **Clean** from the list of Partners.
  - \_\_\_\_f. Select the **Properties** view and the **Description** Tab.
  - \_\_\_\_g. Change the name to AutoClean.
  - \_\_\_h. Select the remaining Partner named **Clean** from the list of the Partners.
  - \_\_\_\_i. Select the **Properties** view and the **Description** tab.

\_\_\_\_j. Change the name to **ManualClean**.



- 4. In order for the Invoke activities to execute requests, they must be set to specific partners. For each Invoke activity, the data for the request message must be specified. This data comes from a variable defined in the business process. For the response on an Invoke, a variable must also be specified to hold the data. For the CleansePublishBPEL process, a single variable can be used as the message type for all request and response messages for all of the services is of type ClipBG. Set the partner for each activity and the variables for the request and response to the inClipBG variable.
  - \_\_\_\_a. Select the AutoClean activity in the Assembly editor and then select Properties view.
  - \_\_\_\_b. Select the **Details** tab and click the **Browse...** button next to the Partner.
  - \_\_\_\_c. Select AutoClean for the Partner.
  - \_\_\_\_d. Click the '...' button for the **Input** variable.
  - \_\_\_\_e. Select InClipBG in the Select Variable for inclipBG window.
  - \_\_\_ f. Click OK.
  - \_\_\_\_g. Click the '...' button for the **Output** variable.
  - \_\_\_\_h. Click **InClipBG** in the Select Variable for outClipBG window.
  - \_\_\_ i. Click **OK**.

Properties 🗙	Problems	Servers	Console	Progress				
Description	🧳 Aul	toClean						
Details	Par <u>t</u> ner:	A	utoClean					Browse
Compensation	Interface	: C	lean					
Join Behavior	Operation	n -	cleanseCli	p			 •	
Correlation	🔽 Use D	ata Tvpe	Variables					
Expiration		/ F	ame	Variable				
Server	🗊 Inpu	t(s) in	ClipBG	InClipBG				
Human Task	Cutr	ut(s) o	utClinBG	InClinBG				
Event Monitor	- odt		accipbo					

**Note**: For the simple CleansePublishBPEL process the same variable can be used as the activities are sequential. For more robust business processes, it might be necessary to have multiple variables in order to reuse original values. If the message types are different, you must use different variables regardless of the sequence of the activities.

- \_\_\_\_j. Select the **ManualClean** Invoke activity in the Assembly editor and then select Properties View.
- \_\_\_\_k. Set the Partner to ManualClean and specify InClipBG for the Input and Output variables.

Properties 🗙	Problems	Servers	Console	Progress							
Description	🔗 ManualClean										
Details	Par <u>t</u> ner:	M	1anualClea	an							(Browse)
Compensation	Interface	: c	lean								
Join Behavior	Operation	Operation: cleanseClip								-	
Correlation	🔽 Use D										
Expiration		- N	ame	Variable							
Server	🗊 Inpu	ıt(s) in	ClipBG	InClipBG							
Human Task	C Outr	out(s) o	utClipBG	InClipBG							
Event Monitor					_						

\_\_\_\_I. Select the **Publish** Invoke activity in the editor.

\_\_\_\_m. Set the Partner to **Publish**. The operation is one-way and only the Input variable needs to be set to InClipBG.

Properties 🗙	Problems Serv	vers Cons	ole Progre	ess					
Description	🧳 Publish	1							
Details	Par <u>t</u> ner:	Publish							( Browse)
Compensation	Interface:	Publish							
Join Behavior	Operation:	publish	Clip			 		 •	•
Correlation	🔽 Use Data T	Voe Variab	les						_
Expiration		Mana	Usviskla						
Server		Name	TeclieRC	_					
Human Task		INCIIPBG	Incliped		•				
Event Monitor									

- 5. Business processes can be short-running or long-running (interruptible). Short-running processes run in a single transaction and are very fast whereas long-running business processes are executed over multiple transactions and can resume after halting and maintain state. In CleansePublishProcess, the ManualClean activity is an asynchronous activity and therefore, CleansePublishProcess must be executed as a long-running business process. In order to indicate to the runtime environment the need to maintain state as the business process waits for the ManualClean activity to complete, CleansePublishProcess must be marked as long-running.
  - \_\_\_\_a. Click on the empty space of the editor to select the entire process and then select Properties tab.
  - \_\_\_ b. Select the **Details** tab.

- \_\_\_\_ c. Select check the box **Process is long-running**.
- \_\_\_\_d. For testing purposes, it is also helpful to have the instance of the business process remain after successful completion. Complete processes can be viewed through the BPC Explorer. Uncheck the box **Automatically delete process after completion**.

Properties 🗙	Problems Servers Console
Description	👤 CleansePublishProcess
Details	Process is long-running
Join Behavior	Automatically delete the process after completion
Imports	Allow optimization
Server	Compensation Sphere: O Supports O Required
Human Task	Autonomy: O Peer O Child
Java Imports	Select date (UTC) when the process becomes valid
Environment	Valid From: Jan V / 01 V / 2002 V
Event Monitor	12 💌 : 00 💌 : 00 💌

\_\_\_\_e. Save the editor (**Ctrl+S**). Check the Problems view for any errors.

## Part 3: Assemble CleansePublish

In this section, you will assemble CleansePublishBPEL, wiring Partner links to targets and define an SCA interface for the business process. As part of the assembly process, the appropriate deployment code will also be generated in preparation for running the business process as a service component on WebSphere Process Server.

- 1. The CleansePublishProcess can be exposed as an SCA component. Add the CleansePublishProcess to the assembly editor.
  - \_\_\_\_a. Expand CleansePublishBPEL in the Business Integration view.
  - \_\_\_\_b. Double-click **CleansePublishBPEL** to open the Assembly editor.
  - \_\_\_\_ c. Drag and drop the CleansePublishProcess which is under Business Logic > Processes onto the Assembly editor from the Business Integration view.



- 2. In the Assembly editor, the partners referenced in the business process can be wired to indicate if the implementation is a local component in the module or exists remotely, outside the module. The interface of the implementation must be added to the Assembly editor and wired to the interfaces (partners) on the CleansePublishProcess. The partners for the CleansePublishProcess are remote SCA components. Wire the partners correctly using Imports.
  - \_\_\_\_a. Expand CleansePublishLibrary and Interfaces in the Business Integration view.
  - \_\_\_\_b. Select the **Clean** interface and drop onto the Assembly Editor.
  - \_\_\_\_ c. Select **Import with No Binding** in the Component Creation window and click **OK**.
  - \_\_\_\_d. With the Import selected on the canvas, select the **Properties** view.
  - \_\_\_\_e. Change the Display name field to **AutoClean** to clarify the assembly editor.

€. •.		<b>D</b> 🕞	NutoClean		
Properties 🗙	Problems	Servers			▼ - □
Description	🕞 Import	: AutoCl	an (No Binding)		
Details	<u>D</u> isplay name:		an	Na <u>m</u> e:	AutoClean
Binding	F <u>o</u> lder:		Move		

- \_\_\_\_\_f. Drop **Clean** onto the Assembly Editor again as the Clean interface is used for both the AutoClean and ManualClean activities even though they rely on different implementations.
- \_\_\_\_g. Select **Import with No Binding** in the Component Creation window and click **OK**.

- \_\_\_h. Change the name to **ManualClean**.
- \_\_\_\_i. Drop **Publish** onto the Assembly Editor.
- \_\_\_\_j. Select Import with No Binding in the Component Creation window and click OK.
- \_\_\_\_k. Change the name to **Publish**.



- \_\_\_\_I. Right-click on the CleansePublishProcess and select **Wire to Existing**. The editor will attempt to wire those partner references to the imports automatically.
- \_\_\_\_\_m. Since the Clean interface is used twice (once for the AutoClean and once for ManualClean), the imports for both outside implementations must be wired to the correct binding. In the Advance Wiring window, select AutoClean under Wire source and check the box for AutoClean in Wire target.

×
g partner reference or target.
Wire target:
🔁 🕞 ManualClean

\_ n. Select **ManualClean** under Wire source and check the box for **ManualClean** in Wire target.

Advanced Wiring				
Multiple autowiring options are available for the followin Select the ones to create:	g partner reference or target.			
Wire source:	Wire target:			
	ManualClean			
ManualClean				

o. Click **OK**. The Publish partner is wired directly to the Publish import.

- \_3. With the partners wired to Imports, the type of binding must be specified. The binding indicates how the implementation should be reached. The implementation for the Imports will be provided by SCA components. For now, you can specify the SCA binding and later select the actual component implementation. Set the binding to SCA.
  - \_\_\_\_a. Right-click on AutoClean and select Generate Binding.... Select SCA Binding.
  - \_\_\_\_b. Right-click on ManualClean and select Generate Binding.... Select SCA Binding.
  - \_\_\_\_ c. Right-click on **Publish** and select **Generate Binding...**. Select **SCA Binding**.



- 4. Besides the different steps of CleansePublishProcess calling SCA components, CleansePublishProcess can also be called as an SCA component. For components (remote) in other modules to call CleansePublishProcess, an Export must be added. Invocation from within the same module can be done by direct wiring without an Export.
  - \_\_\_\_a. Right-click on CleansePublishProcess and select Export... and click SCA Binding.

		🛛 🗟 AutoClean
CleansePublishProcessExport	CleansePublishProcess	-0 🗟 ManualClean
		-0 🗟 Publish

\_\_\_ b. Save the Assembly editor. The appropriate SCA component information will be generated for the CleansePublishProcess.

## Part 4: Unit Test CleansePublish Business Process

In this part, you will unit test the application using the Integration Test Client. Even with unresolved Import component implementations for the partner links, the Integration Test Client allows for the business process logic and requests to be verified prior to integrating with other components. The Integration Test Client will catch outgoing requests and wait until the response is completed and returned.

- \_\_\_\_1. Start the test server.
  - \_\_\_\_\_a. If using a remote testing environment, follow the directions provided in <u>Task: Adding Remote</u> <u>Server to WebSphere Integration Developer Test Environment</u> at the end of this document to add a server to the WebSphere Integration Developer test environment and start it. This is especially true for z/OS, AIX, Solaris remote test environment, where the WebSphere Integration Developer will be remote to the test environment.

If using a local testing environment, change to the Servers view by selecting Servers tab.

\_\_\_\_b. Right-click on WPS Server v6.0. Select Start.

T-Shiel we assess to a super-share

- 2. After the server has restarted, in the Assembly editor, right-click on CleansePublishProcess and select Test Component to test the business process component.
  - \_\_\_\_\_a. A test instance will be opened with a table displaying the initial request parameter which is a business graph (BG). There is a verb for the business graph and the business object, Clip. Fill in the values for the verb and the business object.

Name	Туре	Value		
🖃 inClipBG	ClipBG			
verb	String	Create		
🖃 Clip	Clip			
clipID	string	1000		
GLN	string	1000		
clip	string	testClip		
size	string	500		
color	string	red		
brand	string	dull		
retailItems	ClipItem [ ]	<null></null>		
Data Pool Continue				

\_\_\_\_b. For the **retailItems** value, right-click on the value and select **Add Element**.

ciip	string	cestClip	
size	string 500		
color	string	red	
brand	string	dull	
retailItem:	s ClinItem [ ]	<null></null>	
	Set Value		
Data Pool	Add <u>E</u> lement		Contin <u>u</u> e
	Romovo Flomor		

\_\_\_\_ c. Scroll down and fill in the vales according to the graphic or any values you choose. Again remember to specify values which are compatible with the value type.

Name		Туре	Value	_ ^
	itemID	string	123	
	GTIN	string	456	
	package	string	small	
	quantity	string	1	
	fullDesc	string	dull clips	
	price	double	10.0	
	startDate	date	2005-05-23	=
	endDate	date	2005-06-01	
	contact	string	Jon	
	contact	string	Doe	
	equence	int	0	
<		1111	3	

- \_\_\_\_ 3. Start the test.
  - \_\_\_\_a. Click the **Continue** button under the list of initial request parameters.
  - \_\_\_\_ b. Expand WebSphere Process Server in the Deployment Location window and select WebSphere Process Server v6.0.

👍 Deployment Location	×
Select Deployment Location This server instance is currently running.	
Deployment location:	
WebSphere Process Servers     WebSphere Process Server v6.0	New <u>S</u> er∨er

- \_\_\_\_ c. Click **Finish**. The CleansePublishBPELApp application will be published to the server as well as the Test Connector application which drives the testing on the server.
- 4. The Events window will be updated as the different partner links are invoked by the business process engine. The business process engine will pause and wait for the response for each invoke to be received. The Integration Test Client catches the invoke request and creates a manual emulation which allows the response to be entered.

\_\_\_\_a. The first invoke activity in the CleansePublishProcess is AutoClean which calls the cleanseClip operation. In the Events window, select the Emulate, where the business process engine has paused.



Note: If the Emulate is not shown or an exception is received, restart the server and right-click on Invoke (CleansePublishProcess:cleanseAndPublishClip) and select Rerun.

If using a remote testing environment, stop the server. Right click on WebSphere Process Server v6.0 server from the Servers view and select Stop from the context menu. Then follow the directions provided in <u>Task: Adding Remote Server to WebSphere Integration Developer Test Environment</u> to restart the server.

If using a local testing environment, right click on WebSphere Process Server v6.0 from the Server view and select **Restart** from the context menu.

b. Under the Detailed Properties section, you should see the Input parameters that were sent as the request to the service. The parameters should match the values you specified when starting the test. There should also be a section for the Output parameters. Enter the following values or any values you choose.

Name	Туре	Value
🖃 outClipBG	ClipBG	
verb	String	Create
🖃 Clip	Clip	
clipID	string	2000
GLN	string	2000
clip	string	testClip2
size	string	600
color	string	very red
brand	string	very dull
retailItems	ClipItem [ ]	<null></null>

Output parameters

- \_\_\_\_ c. For the **retailItems** value, right-click on the value and select **Add Element**.
- \_\_\_\_\_d. Expand retailItems and enter the following values or any values you choose.

Output para <u>m</u> eters		
Name	Туре	Value 🔺
🖃 retailItems	ClipItem	
retailItems[0]	ClipItem	
itemID	string	789
GTIN	string	1011
package	string	medium
quantity	string	2
fullDescription	string	very dull clips
price	string	10.0
startDate	string	2005-06-01
endDate	string	2005-07-15
contactFirstName	string	Jane
contactLastName	strina	Smith

\_\_\_\_\_e. Before continuing the test, save the parameters you just entered into the data pool for reuse. Scroll to the top of the Output parameters and right-click on ClipBG and select Add Value to Pool.

Output parameters				
Name	Туре		Value	<b></b>
outClipBG	Cli	Set V	alue	
verb	Str	500 1		
🖃 Clip	Clij	Add B	lement	
clipID	str			
GLN	str	<u>R</u> ema	ve Element	
clip	str	Con	Valua	
size	str	Сору	value	
color	str	Add \	al <u>u</u> e to Pool	
brand	str	Use V	alue <u>f</u> rom Pool	
retailItems	Clij	Paste	Value	
retailItems[0]	ClipIte	em		
itemID	strina		789	•

- \_\_\_\_\_f. Accept the default value name and click **OK**.
- \_\_\_\_g. Click the **Continue** button to send the response back to the business process engine.
- \_\_\_\_h. The response will be returned to the business process engine and execution will continue until the next invoke, ManualClean is reached. This activity also uses the cleanseClip interface. The test will wait until the response is specified. Notice the Input parameters are from the values you entered previously.



\_\_\_\_i. Fill out the Output parameters by selecting the ClipBG and selecting Use Value from Pool...

Output para <u>m</u> eters		
Name	Туре	Value
outClipBG	ClipBG	Set Value
verb	String	
🖃 Clip	Clip	Add Element
clipID	string	
GLN	string	<u>R</u> emove Element
clip	string	Copy Value
size	string	Copy value
color	string	Add Val <u>u</u> e to Pool
brand	string	Use Value from Pool
retailItems	ClipItem [ ]	Paste Value

- - \_\_\_\_j. Select **inClipBG** from the list of values and click **OK**. The Output parameters will be populated with the values from the pool and can be easily reused or changed.
  - \_\_\_\_k. Under outClipBG and Clip, change clipID and GLN to 3000.
  - \_\_\_\_I. Click **Continue**. The response is returned and the business process will continue.

Events
- 🐩 Invoke (CleansePublishProcess:cleanseAndPublishClip)
🚊 🖓 Started
Invoke (CleansePublishProcess:cleanseAndPublishClip
Response (CleansePublishProcess < AutoClean:clea
Response (CleansePublishProcess < ManualClean:c
Request (CleansePublishProcess> Publish:publishC
Pri Emulate (Publish:publishClip)
Stopped

\_\_\_\_m. The final invoke, Publish, is reached. You can see the input parameters which has been passed to the Publish service. Since the invoke is a one-way operation, there is no response message or output parameters to specify.

Input <u>p</u> arameters			
Name	Туре	Value 🔺	
🖃 inClipBG	ClipBG		
verb	VerbType	Create	
🖃 Clip	Clip		
clipID	ClipIDType	3000	
GLN	GLNType	3000	
clip	ClipType	testClip2	
size	SizeType	600	
color	String	very red	
brand	BrandType	very dull	
🖃 retailItems	ClipItem [ ]		
eretailItem	ClinItem		
Output parameters			
Name	Туре	Value	

- \_\_\_\_n. Click **Continue** to complete the request. The business process will complete and the test will end.
- \_\_\_\_5. Clean up test environment.
  - \_\_\_\_a. Right-click on appropriate WPS Server on which you deployed the applications in the Servers view and select Add and remove projects...
  - \_\_\_ b. Click << Remove All.
  - \_\_\_ c. Click Finish.
- 6. Stop the Server. Right click on WebSphere Process Server v6.0 server from the Servers view and select Stop from the context menu.

## What you did in this exercise

In this exercise, you built a business process that follows the WSBPEL specification. The business process you built contained Invoke activities that called 3 different SCA components, defined by Partner links. Because these components are located in different modules, you wired the interfaces for the Partner links to SCA components using Imports. Finally you component tested the business process using the Integration Test Client.

## **Solution Instructions**

1. Follow the directions in the task <u>Initialize the Workspace for a Lab Exercise</u>, using the following values:

#### <WORKSPACE>

C:\Labfiles60\eXchange\CleanPublishBPEL\workspace

#### <PROJECT\_INTERCHANGE>

 $C:\Labfiles 60\exchange\Clean Publish BPEL\solution\Clean Publish BPEL_PI.zip$ 

#### <MODULE>

n/a

#### <DEPENDENT\_LIBRARIES>

n/a

#### 2. Continue with Part 4: Unit Test CleansePublish Business Process.

# Task: Adding Remote Server to WebSphere Integration Developer Test Environment

This task describes how to add a remote server to the WebSphere Integration Developer Test environment. This sample will use a z/OS machine.

Create a new remote server.

- 1. Right click on the background of the Servers view to access the pop-up menu.
- 2. Select **New > Server.**

Properties Problems 👫 Servers 🗙 Console		🌣 🕥 🤣	🍫 🔲 🙌 📪 🗖	
Server	Host name	Status	State	
HebSphere ESB Server v6.0	localhost	🖥 Stopped	Synchronized	
WebSphere Process Server v6.0	localhost	🖥 Stopped	Synchronized	
New Server				

- 3. Specify hostname to the remote server, <HOSTNAME>.
- 4. Ensure that 'WebSphere Process v6.0 Server' is highlighted in the server type list.

🚯 New Server	×
Define a New Server Choose the type of server to create.	
Specify the host where you want to publish	
Host name: mvsxxx.rtp.raleigh.ibm.com	•
Select the server type:	
IBM         WebSphere ESB Server v6.0         WebSphere Express v5.0 Server         WebSphere Express v5.1 Server         WebSphere Process v6.0 Server         WebSphere v5 Server Attach	
View By: Vendor Description: WebSphere Process v6.0 Server	•
k	

- 5. Click Next.
- 6. On the WebSphere Server Settings page, select the radio button for **RMI** and change the ORB bootstrap port to the correct setting (<**BOOTSTRAP\_PORT**>).

🔂 New Server	×
WebSphere Server Settings	
Input settings for the new WebSphere server]	
WebSphere profile name:	<b>v</b>
Server connection type and admin port	
• RMI (Better performance)	
ORB bootstrap port: 9131	
C SOAP (More firewall compatible)	
SOAP connector port: 8880	
Pup correct with reconnect within the workspace	
Security is enabled on this server	
Current active authentication settings:	
User ID:	
Pacemord	
Server type	
BASE, Express or unmanaged Network Deployment server	
C Network Deployment server	
Network Deployment server name:	
The server name is in the form of: <cell name="">/<node name="">/<server name=""> For example, localhost/localhost/server1.</server></node></cell>	
Detect Click this button to detect the server type.	
< Back Next > Finish	Cancel

#### 7. Click **Finish**.

8. The new server should be seen in the Server view.

Properties Problems 🛠 Servers 🗙 Console		🌣 🕥 🖉	🤣 🔳 🙌 📪 🗖
Server	Host name	Status	State
🛅 WebSphere ESB Server v6.0	localhost	🖥 Stopped	Synchronized
WebSphere Process Server v6.0	localhost	🖥 Stopped	Synchronized
WebSphere Process v6.0 Server @ mvsxxx.rtp.ral	mvsxxx.rtp.raleigh.ibm.com	Started	Synchronized
<u>▼</u>			

Start the remote server if it is not already started. WebSphere Integration Developer does not support starting remote servers from the Server View.

9. From a command prompt, telnet to the remote system if needed:

#### 'telnet <HOSTNAME> <TELNET\_PORT>'

userid : <USERID>

pw: <PASSWORD>

10. Navigate to the bin directory for the profile being used:

#### cd <WAS\_HOME>/profiles/<PROFILE\_NAME>/bin

- 11. Run the command file to start the server: ./startServer.sh <SERVER\_NAME>
- 12. Wait for status message indicating server has started:

ADMU3200I: Server launched. Waiting for initialization status.

ADMU3000I: Server cllsr01 open for e-business; process id is 000001200000002