IBM WebSphere® process integration 6.0.2 – Lab exercise

CleansePublish BPEL

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 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 V6.0.2 installed
- WebSphere Process Server V6.0 test environment installed
- Sample code in the directory C:\Labfiles602 (Windows®) or /tmp/LabFiles602 (Linux®)

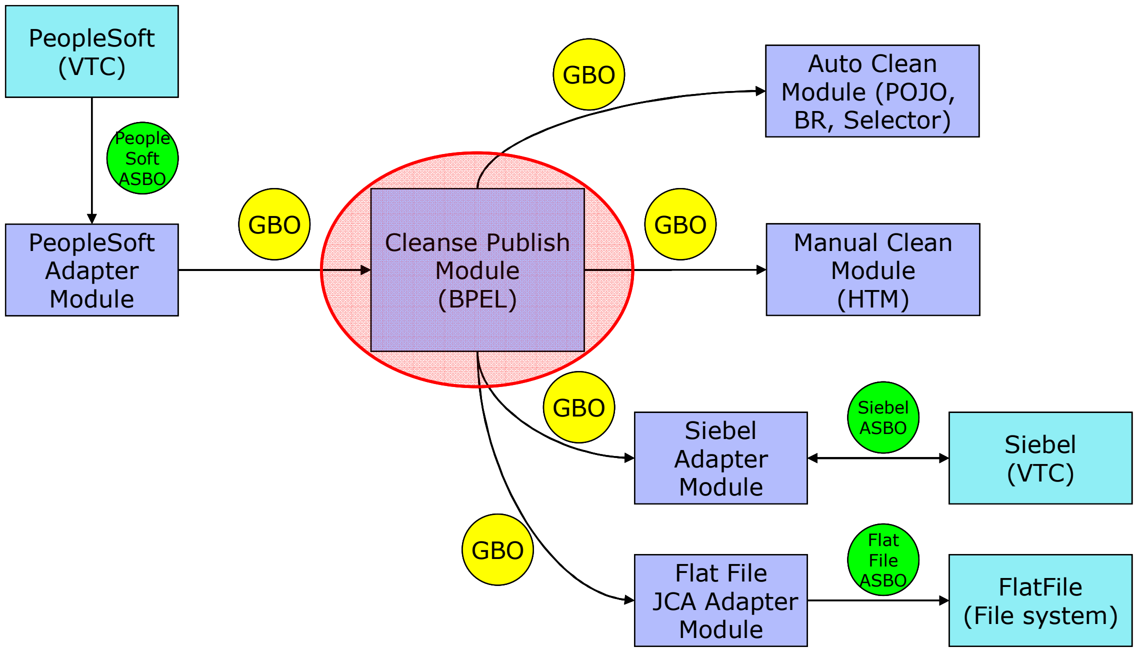
What you should be able to do

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

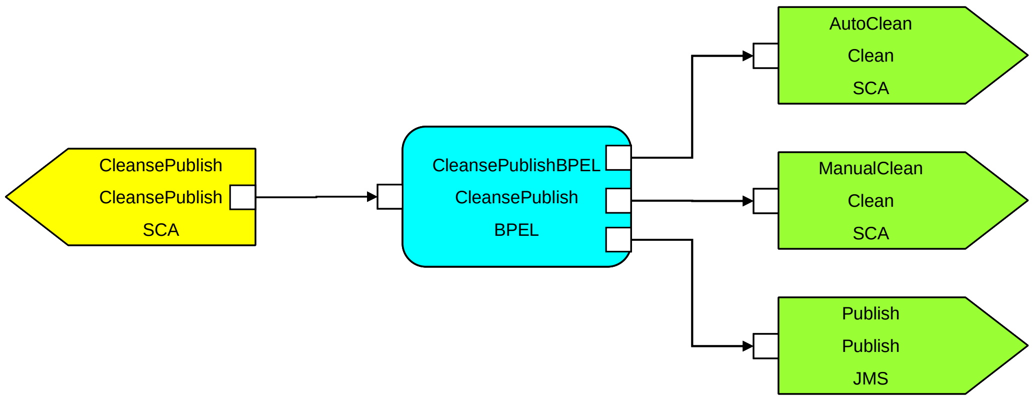
Introduction

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™ 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.


PeopleSoft (VTC)
People Soft ASBO
PeopleSoft Adapter Module
GBO
Cleanse Publish Module (BPEL)
GBO
Auto Clean Module (POJO, BR, Selector)
GBO
Manual Clean Module (HTM)
GBO
Siebel Adapter Module
Siebel ASBO
Siebel (VTC)
GBO
Flat File JCA Adapter Module
Flat File ASBO
FlatFile (File system)

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>	C:\Program Files\IBM\WebSphere\ID\6.0	/opt/IBM/WebSphere/ID/6.0
<WPS_HOME>	<WID_HOME>\runtimes\bi_v6	<WID_HOME>/runtimes/bi_v6
<LAB_FILES>	C:\Labfiles602	/tmp/Labfiles602
<WORKSPACE>	C:\Labfiles602\eXchange\CleanPublishBPEL\ workspace	/tmp/Labfiles602/eXchange/ CleanPublishBPEL/workspace
<TEMP>	C:\temp	/tmp
<SOLUTION>	C:\Labfiles602\CleansePublishBPEL\Solution	/tmp/Labfiles602/CleansePublishBP EL/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:\LabFiles602\ would be replaced by C:/LabFiles602/.

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

Reference Variable	**Example: Remote Windows test server location**	**Example: Remote z/OS test server location**	**Input your values for the remote location of the test server**
<SERVER_NAME>	server1	cl1sr01	
<WAS_HOME>	C:\Program Files\IBM\WebSphere\AppServer	/etc/cl1cell/AppServerNode1	
<HOSTNAME>	localhost	mvsxxx.rtp.raleigh.ibm.com	
<PORT>	9080	9080	
<BOOTSTRAP_PORT>	2809	2809	
<TELNET_PORT>	N/A	1023	
<PROFILE_NAME>	AppSrv01	default	
<USERID>	N/A	cl1admin	
<PASSWORD>	N/A	fr1day	

Instructions for using a remote testing environment, such as z/OS®, AIX® or Solaris, can be found at the end of this document, in the section "**Task: Adding remote server to WebSphere Integration Developer test environment**".

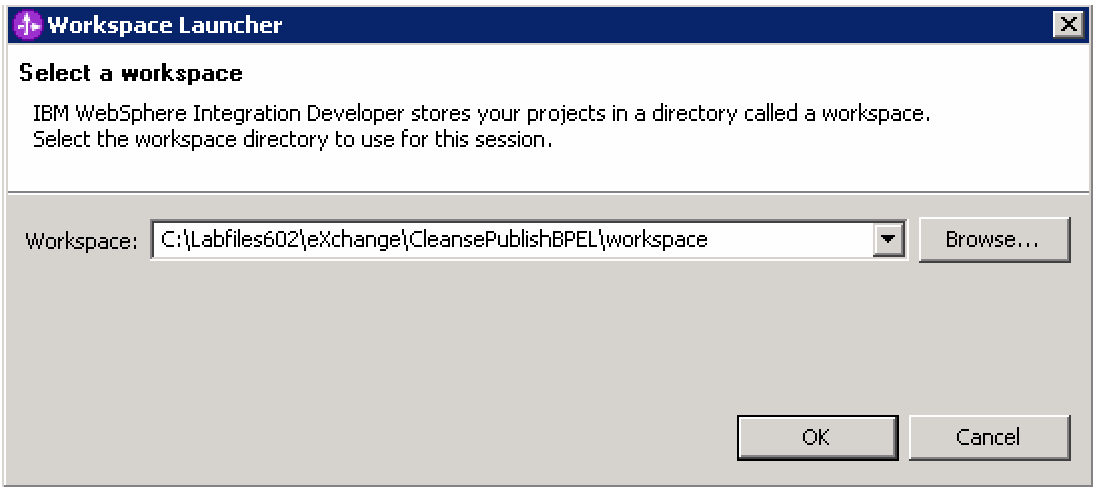
Part 1: Initialize the workspace for this lab exercise

In this section of the lab, you will be importing a library project part of the CleansePublishLibrary_PI.zip project interchange file into your workspace and eventually create a new module named CleansePublishBPEL.

____ 1. Start WebSphere Integration Developer V6.0.2 with a workspace location of **<WORKSPACE>**, that is **<LAB_FILES>\eXchange\CleansePublishBPEL\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**

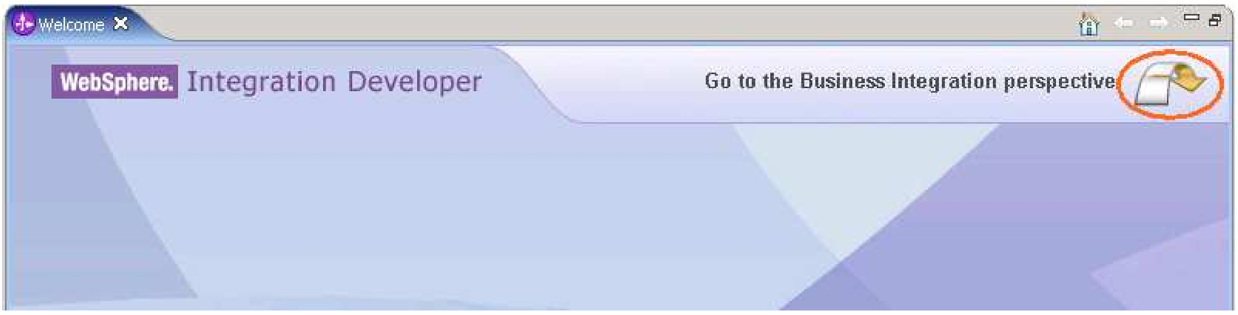


Select a workspace

IBM WebSphere Integration Developer stores your projects in a directory called a workspace.
Select the workspace directory to use for this session.

Workspace: C:\Labfiles602\eXchange\CleansePublishBPEL\workspace

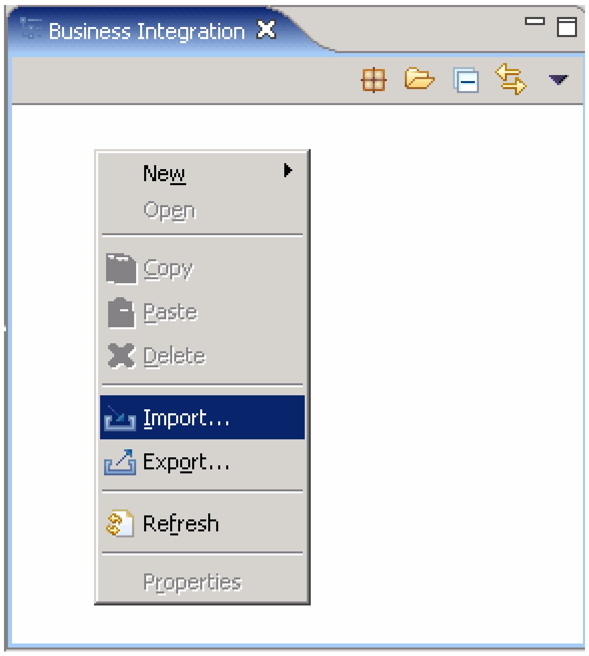
__ c. When WebSphere Integration Developer V6.0.2 opens, click the curved arrow at top right to **go to Business Integration perspective**


Welcome
WebSphere. Integration Developer
Go to the Business Integration perspective

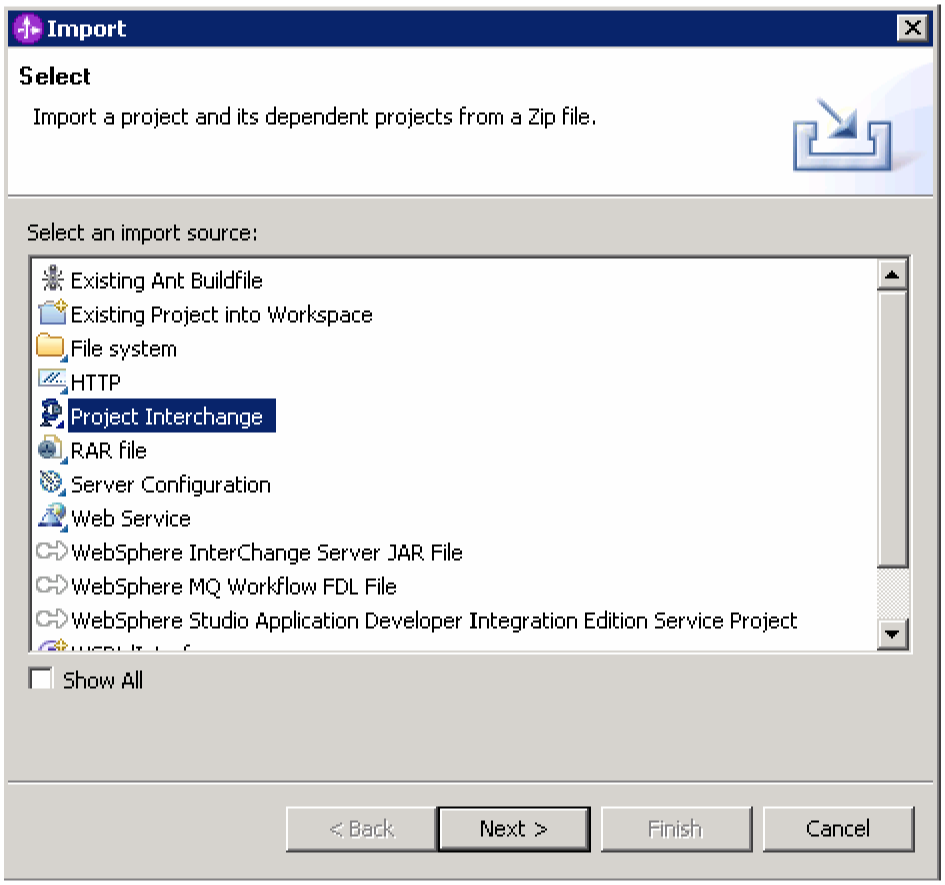
__ d. Ensure you are in the **Business Integration** perspective

____ 2. Import Project Interchange file, **CleansePublishLibrary_PI.zip** located at **<LAB_FILES>\eXchange\CleansePublishLibrary\import**

__ a. Right-click inside **Business Integration view** (top left view in the Business Integration perspective) and select **Import** from the context menu

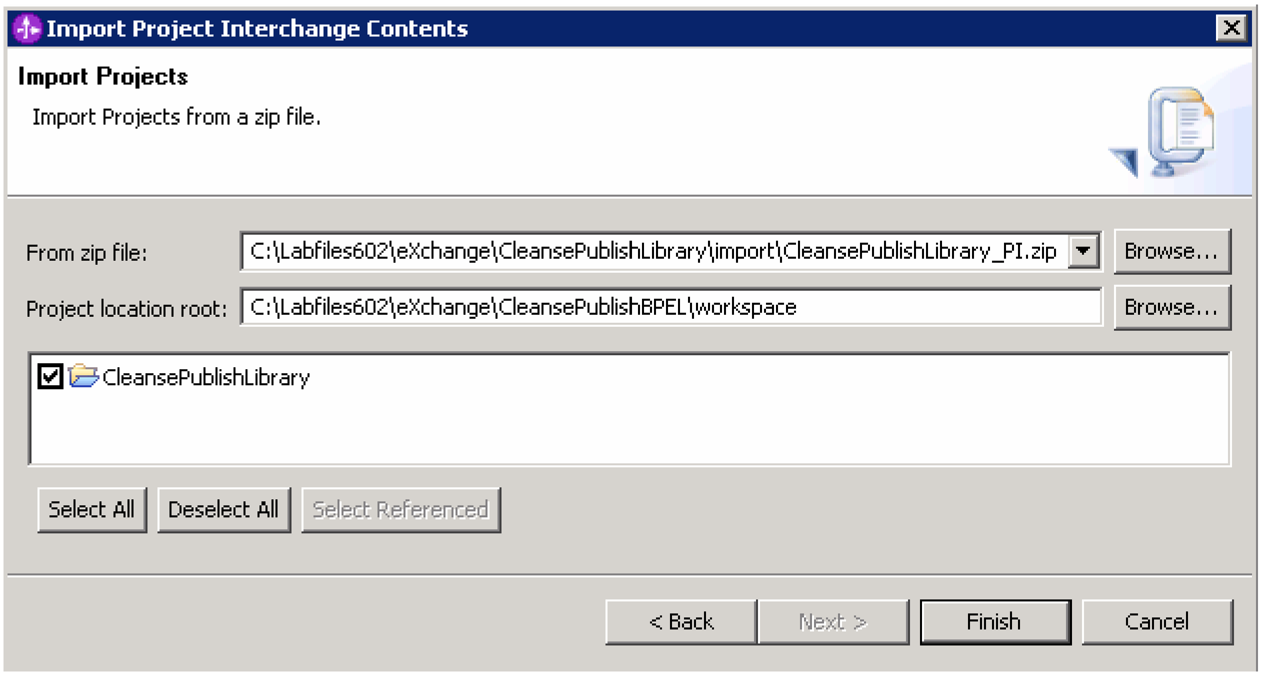


__ b. Select **Project Interchange** listed in the import dialog


Import
Select
Import a project and its dependent projects from a Zip file.
Select an import source:
Existing Ant Buildfile
Existing Project into Workspace
File system
HTTP
Project Interchange
RAR file
Server Configuration
Web Service
WebSphere InterChange Server JAR File
WebSphere MQ Workflow FDL File
WebSphere Studio Application Developer Integration Edition Service Project
Show All
< Back
Next >
Finish
Cancel

__ c. Click **Next**

__ d. Click the **Browse** button for "**From zip file**" and navigate to **<LAB_FILES>\eXchange\CleansePublishLibrary\import\CleansePublishLibrary_PI.zip** and hit **Open**


Import Project Interchange Contents

Import Projects

Import Projects from a zip file.

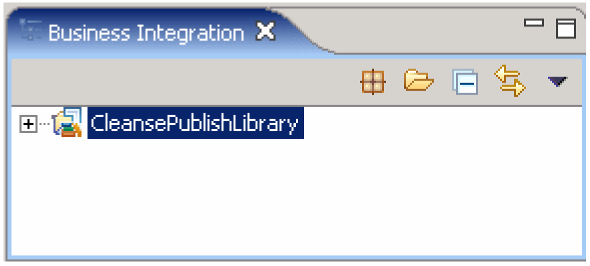
From zip file: C:\Labfiles602\eXchange\CleansePublishLibrary\import\CleansePublishLibrary_PI.zip

Project location root: C:\Labfiles602\eXchange\CleansePublishBPEL\workspace

☑ CleansePublishLibrary

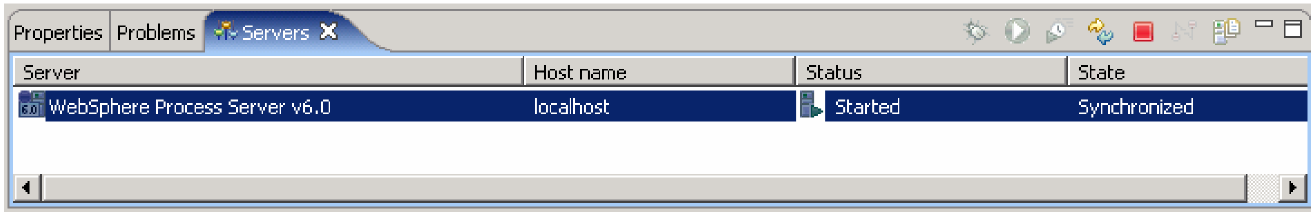
__ e. Select the check box next to **CleansePublishLibrary** and click **Finish**

__ f. Verify you have **CleansePublishLibrary** module listed in the Business Integration view



CleansePublishLibrary

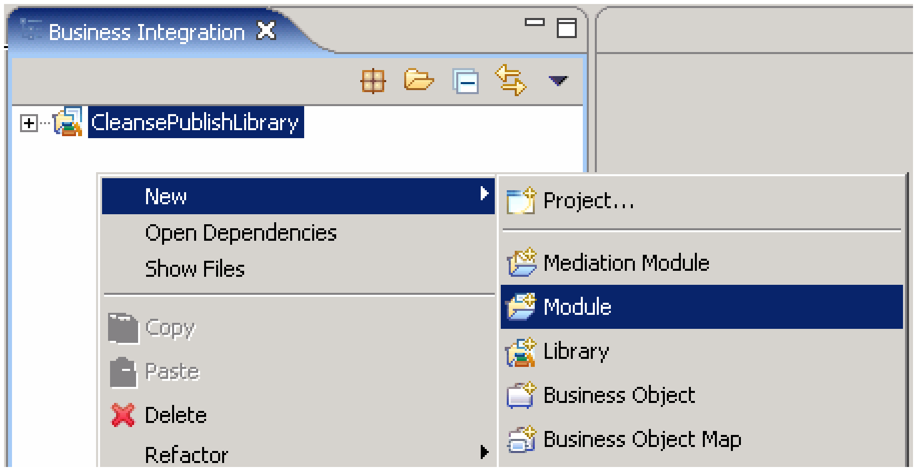
__ g. Verify you have WebSphere Process Server V6.0 listed in your Servers view



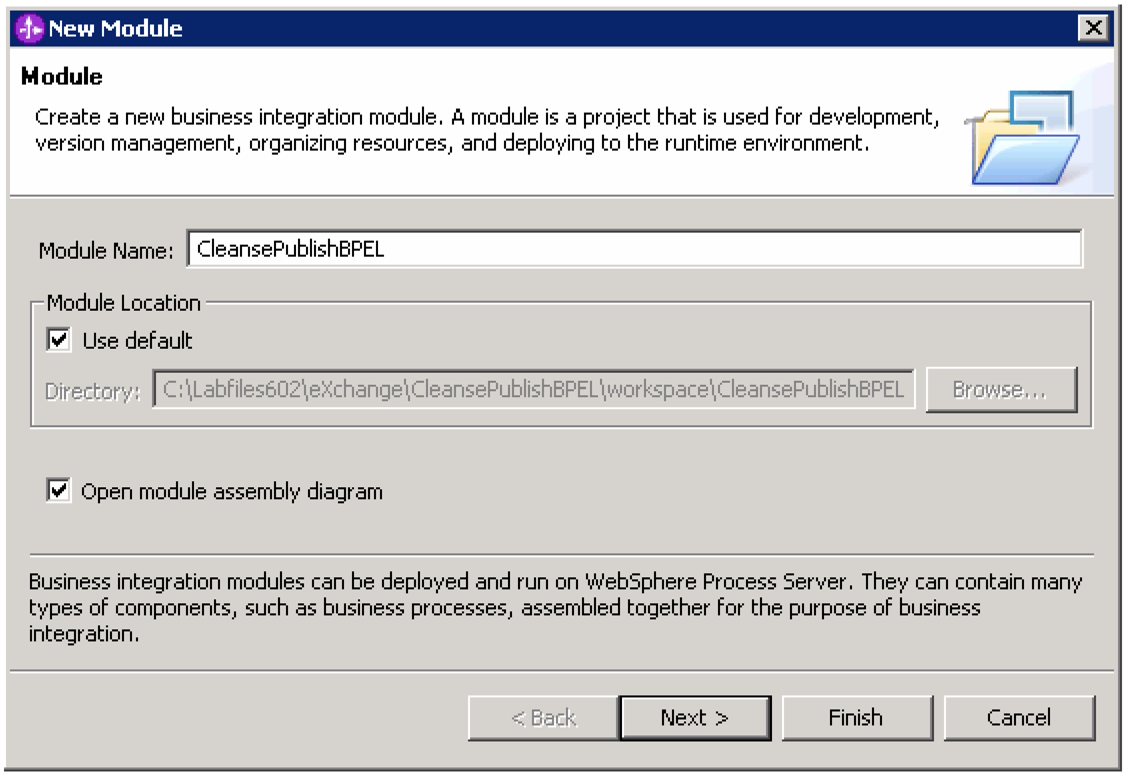
Server	Host name	Status	State
WebSphere Process Server v6.0	localhost	Started	Synchronized

____ 3. Create a new module named, **CleansePublishBPEL**

__ a. Right-click in Business Integration view and select **New → Module**



__ b. Enter the Module Name as **CleansePublishBPEL**


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

[✓] Use default

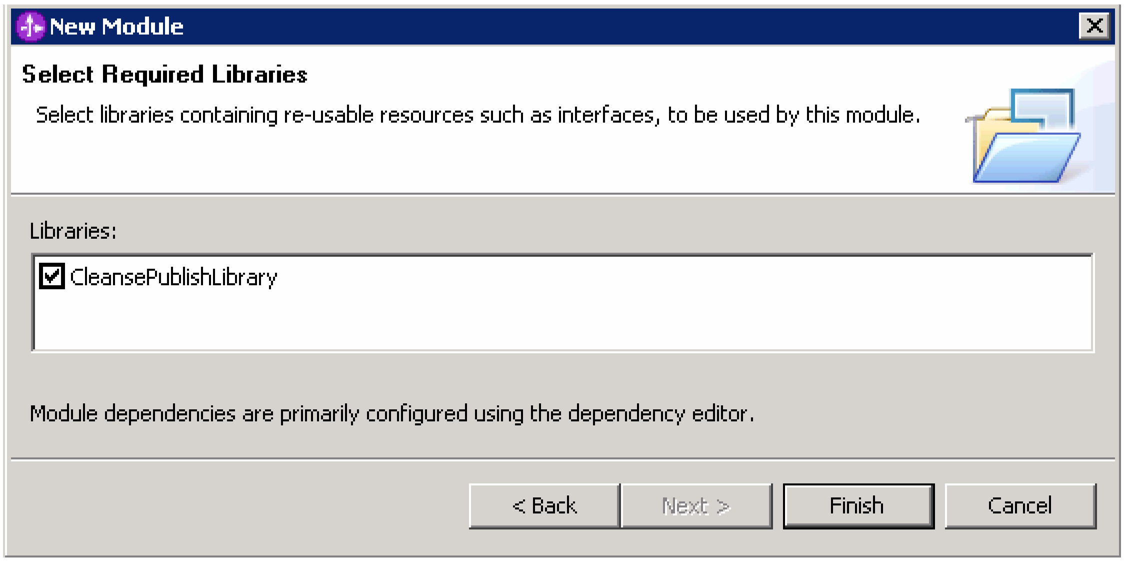
Directory: C:\Labfiles602\eXchange\CleansePublishBPEL\workspace\CleansePublishBPEL

[✓] Open module assembly diagram

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.

__ c. Click **Next**

__ d. In the following panel, select the check box next to **CleansePublishLibrary**. This adds the CleansePublishLibrary as a dependant library for the CleansePublishBPEL module



Select Required Libraries

Select libraries containing re-usable resources such as interfaces, to be used by this module.

Libraries:

☑ CleansePublishLibrary

Module dependencies are primarily configured using the dependency editor.

__ e. Click **Finish**

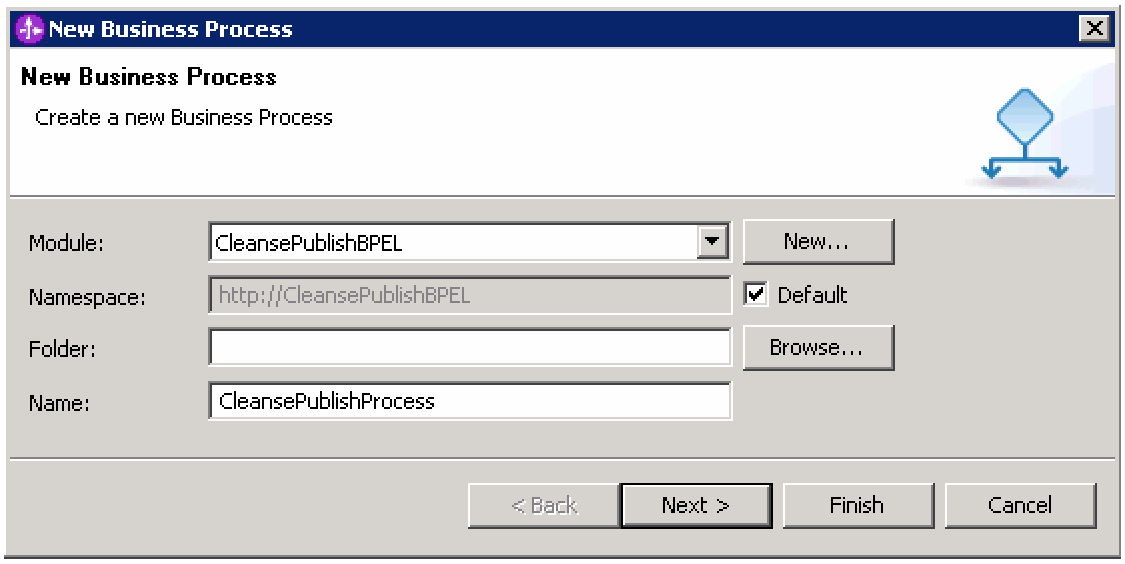
Part 2: Construct CleansePublish BPEL business process

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

____ 1. Create an empty business process named **CleansePublishBPEL**

__ a. In the Business Integration view, expand **CleansePublishBPEL → Business Logic,** right click on **Processes** and select **New → Business Process** from the context menu

__ b. Enter **CleansePublishProcess** for the Name filed


New Business Process

New Business Process

Create a new Business Process

Module: CleansePublishBPEL

Namespace: http://CleansePublishBPEL

[✓ Default]

Folder:

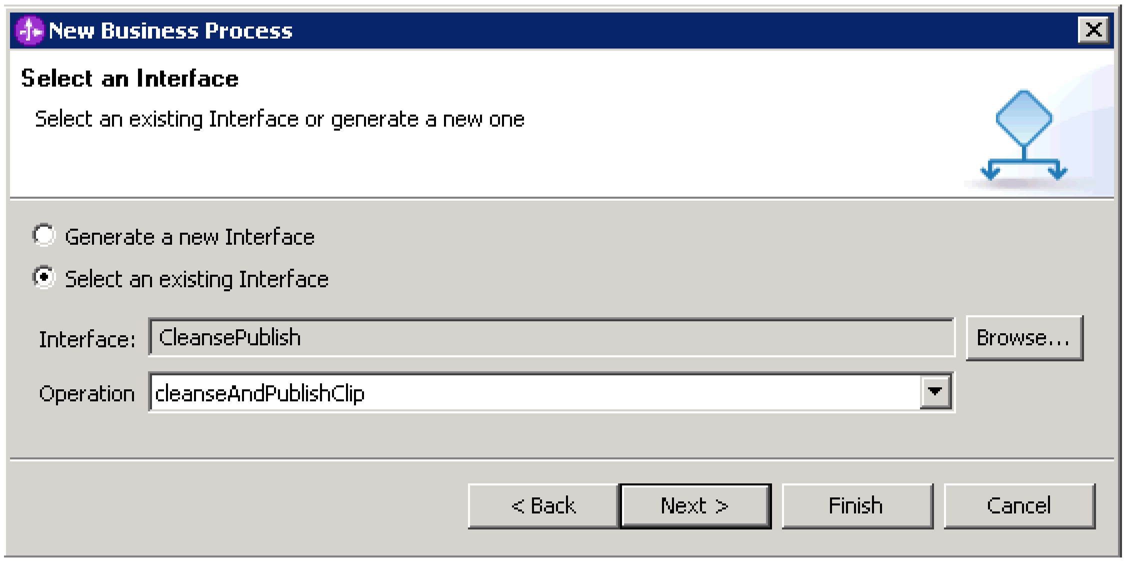
Name: CleansePublishProcess

__ c. Click **Next**

__ d. In the following panel, select the radio button next to **Select an Existing Interface**

__ e. Click the **Browse…** button for Interface and select **CleansePublish** from the list

__ f. You will see the Operation populated with "**cleanseAndPublishClip**"

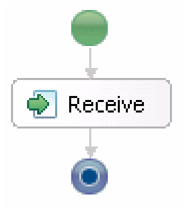


( ) Generate a new Interface
(●) Select an existing Interface

Interface: CleansePublish

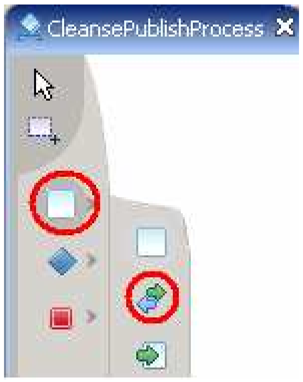
Operation: cleanseAndPublishClip

__ g. Click **Finish**. The business process will be created and opened in the editor

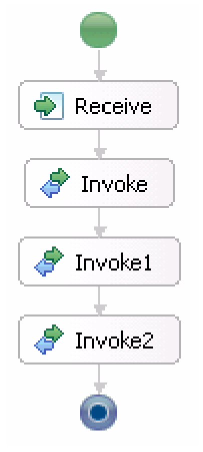


____ 2. The CleansePublishProcess is a simple business process composed of a series of steps that form the business logic. Define the business logic with Invoke activities

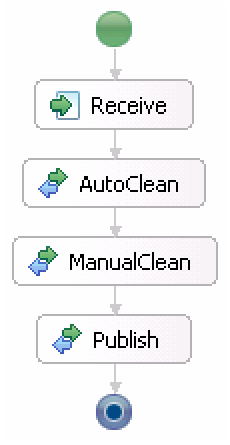
__ a. Select the **Invoke** activity from the activity palette on the left side of the editor



__ b. Drop 3 Invoke activities under the **Receive** activity. The Invoke activities carry out requests to different service components to run different parts of the business process



__ c. Rename the activities to **AutoClean**, **ManualClean**, and **Publish** by selecting the activity and then the **Description** tab under **Properties**



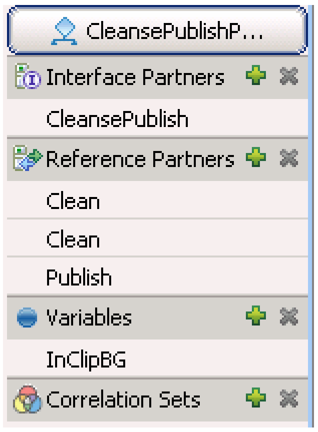
____ 3. The various steps in the business process need services to call. These services are defined by interfaces and called partner links. Create the partner links for the process.

__ a. In the Business Integration view, expand **CleansePublishLibrary → Interfaces**

__ b. Drag **Clean** onto the process editor. You should see the Interface appear under **Reference Partners** tab located to the right of the Process Diagram

__ c. Drag **Clean** onto the process editor again

__ d. Drag **Publish** onto the process editor

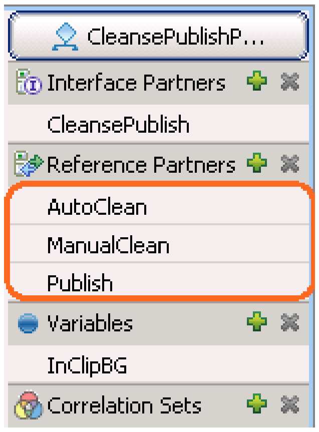


__ e. Select the first partner listed, that is **Clean** from the list of Reference Partners and then the **Description** tab in the Properties view

__ f. Change the name to **AutoClean**

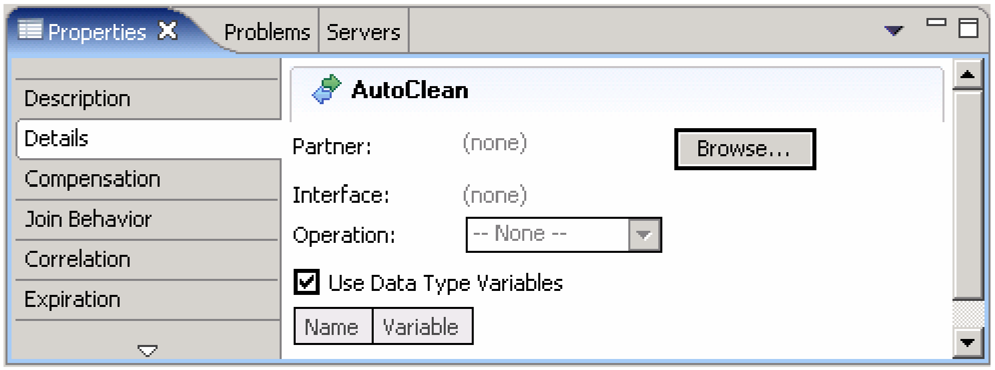
__ g. Select the remaining Partner named **Clean** from the list of the Reference Partners and then the **Description** tab in the Properties view

__ h. Change the name to **ManualClean**



____ 4. In order for the Invoke activities to run 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



Partner: (none)
Interface: (none)
Operation: -- None --
☑ Use Data Type Variables

__ b. Select the **Details** tab and click the **Browse…** button next to the **Partner**

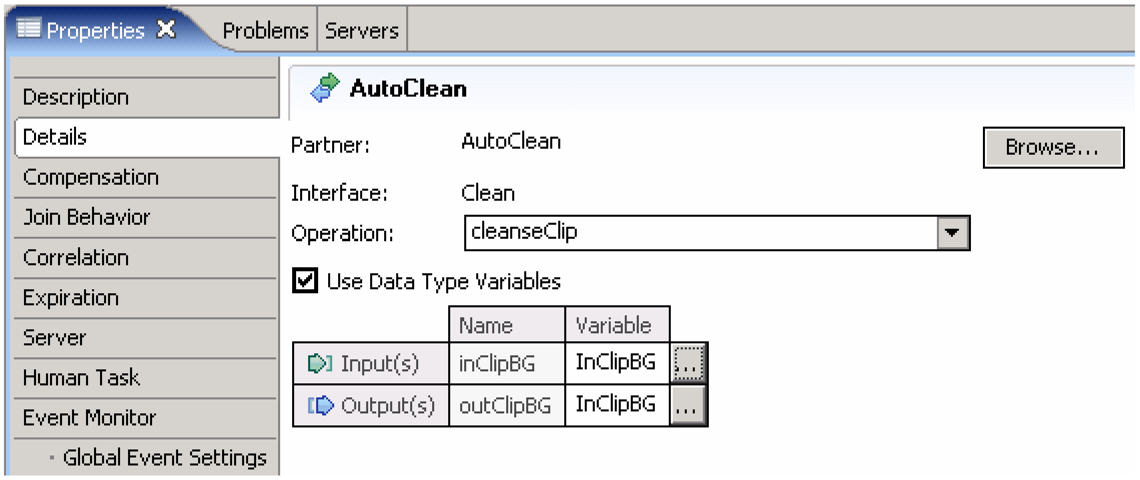
__ c. Select **AutoClean** from the Select a Partner dialog and click **OK**

__ d. Click the (…) button for the **Input** variable

__ e. Select **InClipBG** in the Select Variable for inClipBG dialog and click **OK**

__ f. Now click the (…) button for the **Output** variable

__ g. Select **InClipBG** in the Select Variable for outClipBG dialog and click **OK**

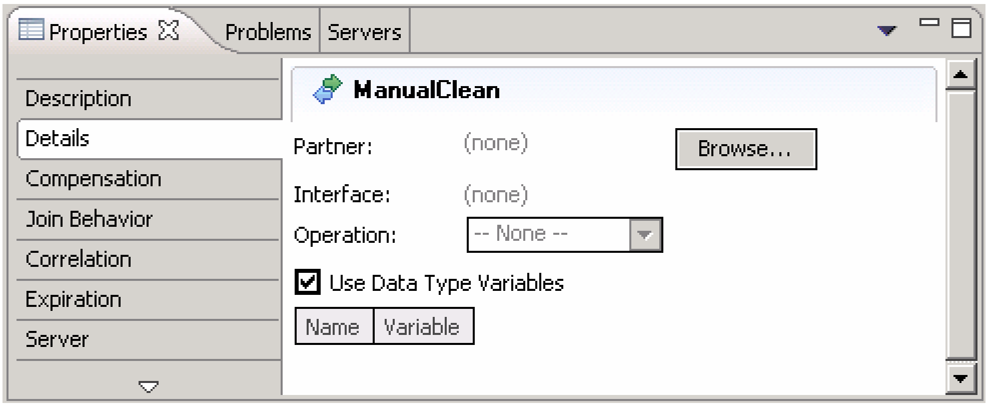


Partner: AutoClean
Interface: Clean
Operation: cleanseClip
☑ Use Data Type Variables

	Name	Variable
Input(s)	inClipBG	InClipBG
Output(s)	outClipBG	InClipBG

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.

__ h. Select the **ManualClean** Invoke activity in the Assembly editor and then select Properties View



ManualClean

Partner: (none)

Interface: (none)

Operation: -- None --

☑ Use Data Type Variables

Name	Variable

__ i. Select the **Details** tab and click the **Browse…** button next to the **Partner**

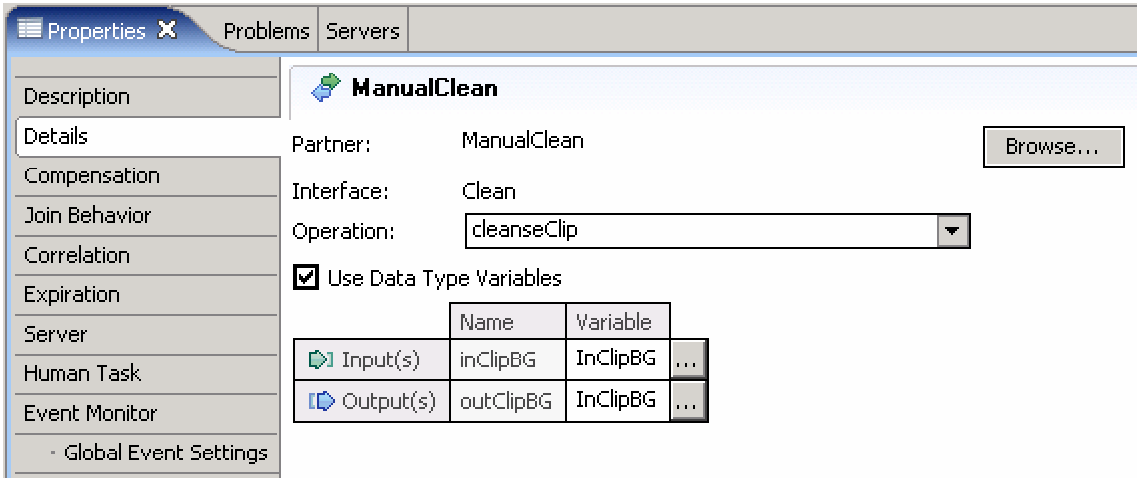
__ j. Select **ManualClean** from the Select a Partner dialog and click **OK**

__ k. Click the (…) button for the **Input** variable

__ l. Select **InClipBG** in the Select Variable for inClipBG dialog and click **OK**

__ m. Now click the (…) button for the **Output** variable

__ n. Select **InClipBG** in the Select Variable for outClipBG dialog and click **OK**



ManualClean

Partner: ManualClean

Interface: Clean

Operation: cleanseClip

☑ Use Data Type Variables

	Name	Variable	
Input(s)	inClipBG	InClipBG	...
Output(s)	outClipBG	InClipBG	...

__ o. Select the **Publish** Invoke activity in the Assembly editor and then select Properties View

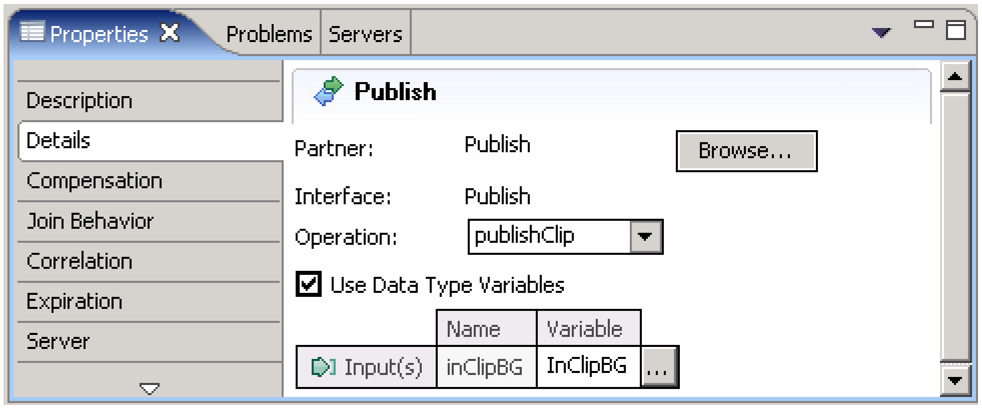
__ p. Select the **Details** tab and click the **Browse…** button next to the **Partner**

__ q. Select **Publish** from the Select a Partner dialog and click **OK**

__ r. Click the (…) button for the **Input** variable

__ s. Select **InClipBG** in the Select Variable for inClipBG dialog and click **OK**

Note: The operation is one-way and only the Input variable needs to be set to InClipBG.



Partner: Publish
Interface: Publish
Operation: publishClip
[✓] Use Data Type Variables

	Name	Variable
Input(s)	inClipBG	InClipBG

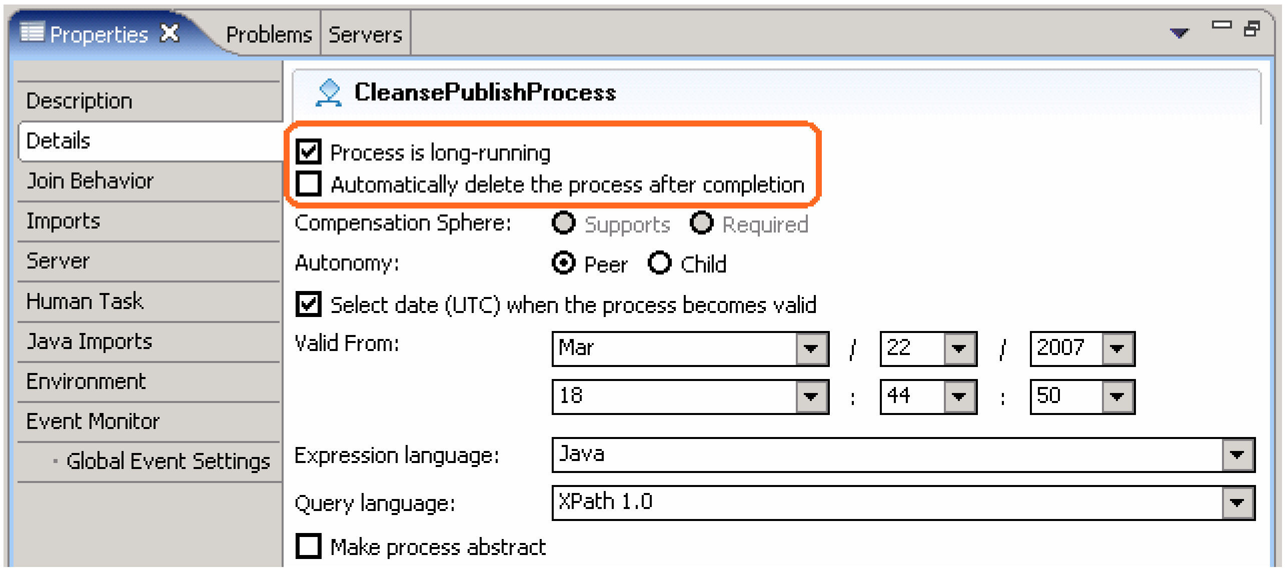
____ 5. Business processes can be short-running or long-running (interruptible). Short-running processes run in a single transaction and are very fast whereas long-running business processes are run over multiple transactions and can resume after halting and maintain state. In CleansePublishProcess, the ManualClean activity is an asynchronous activity and therefore, CleansePublishProcess must be run 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 white canvas (empty space) of the editor to select the entire process and then select Properties tab

__ b. Select the **Details** tab

__ c. Select the check box next to **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. Unselect the check box next to **Automatically delete process after completion**



[✓] Process is long-running
[ ] Automatically delete the process after completion
Compensation Sphere: ○ Supports ○ Required
Autonomy: [✓ Peer] ○ Child
[✓] Select date (UTC) when the process becomes valid
Valid From: Mar / 22 / 2007
18 : 44 : 50
Expression language: Java
Query language: XPath 1.0
[ ] Make process abstract

__ e. Save the editor (**Ctrl+S**). Check the Problems view for any errors

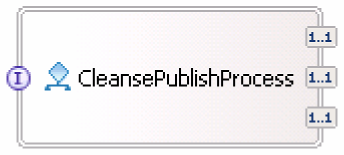
Part 3: Assemble CleansePublish

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.

____ 6. The CleansePublishProcess can be exposed as an SCA component. Add the CleansePublishProcess to the assembly editor.

__ a. Expand **CleansePublishBPEL** in the Business Integration view and double click on **Assembly Diagram** (Assembly Diagram) to open it in an Assembly Diagram editor

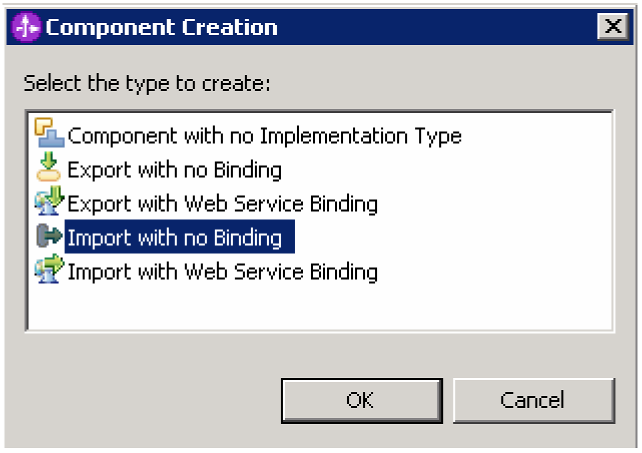
__ b. Expand **CleansePublishBPEL → Business Logic → Processes**, drag the **CleansePublishProcess** onto the Assembly editor



____ 7. 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. In the Business Integration view, expand **CleansePublishLibrary → Interfaces;** drag the **Clean** interface onto the Assembly Editor. The Component Creation dialog pops-up

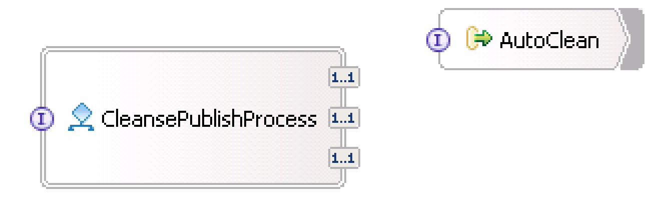
__ b. In the Component Creation dialog, select **Import with No Binding**


Component Creation
Select the type to create:
Component with no Implementation Type
Export with no Binding
Export with Web Service Binding
Import with no Binding
Import with Web Service Binding
OK
Cancel

__ c. Click **OK**

__ d. With the Import selected on the Assembly diagram, select the **Properties** view

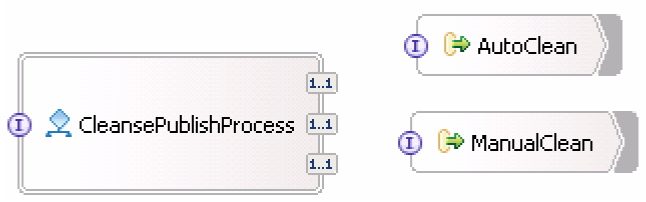
__ e. Change the Display name field to **AutoClean**



__ f. Drop the **Clean** Interface 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 dialog and click **OK**

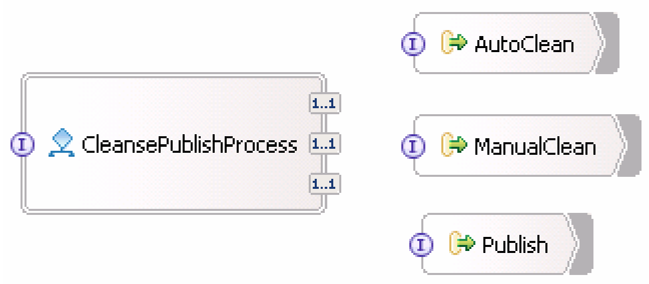
__ h. Change the name to **ManualClean**



__ i. Now drop the **Publish** Interface onto the Assembly Editor

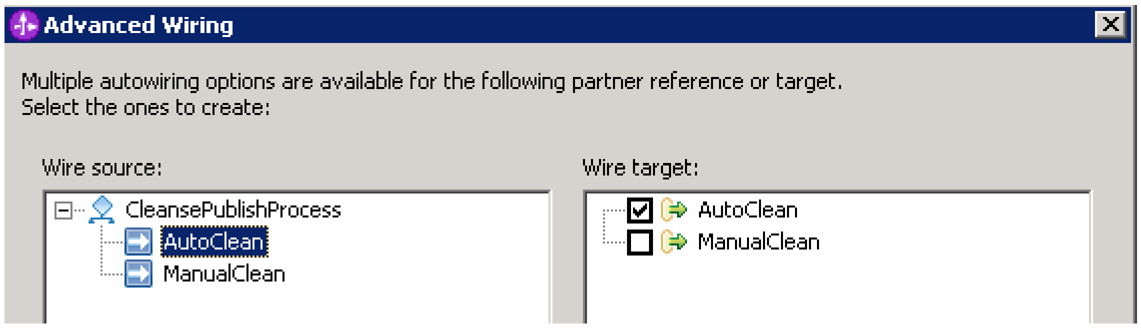
__ j. Select **Import with No Binding** in the Component Creation dialog and click **OK**

__ k. Change the name to **Publish**



__ l. Right-click on the CleansePublishProcess and select **Wire to Existing** from the context menu. 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** panel, select **AutoClean** under Wire source and select the check box next to **AutoClean** in Wire target

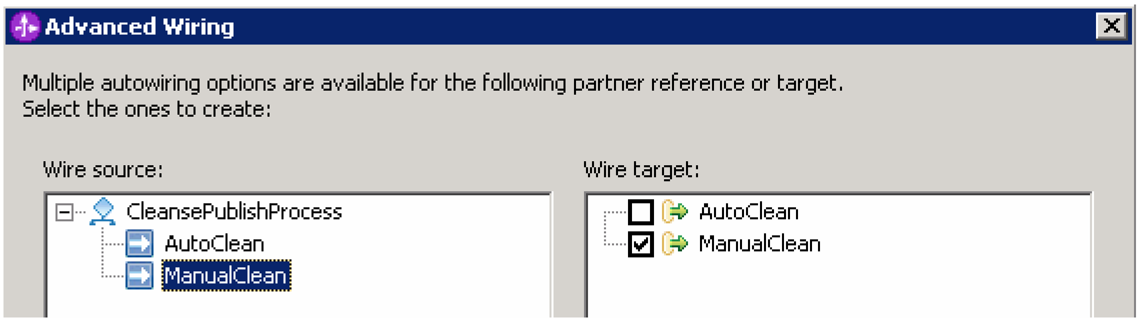


Multiple autowiring options are available for the following partner reference or target.
Select the ones to create:

Wire source:
CleansePublishProcess
AutoClean
ManualClean

Wire target:
[✓] AutoClean
[ ] ManualClean

__ n. Now select **ManualClean** under Wire source and select the check box next to **ManualClean** in Wire target

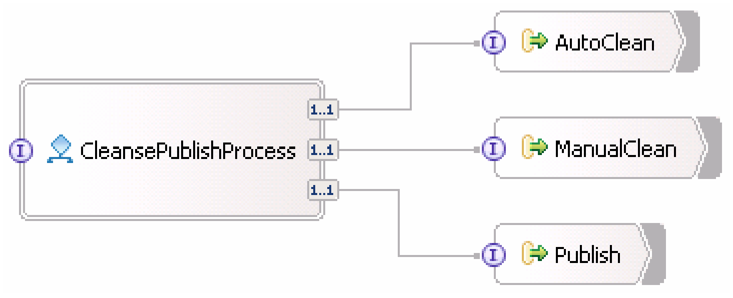

Advanced Wiring

Multiple autowiring options are available for the following partner reference or target.
Select the ones to create:

Wire source:
- CleansePublishProcess
  - AutoClean
  - ManualClean

Wire target:
- [ ] AutoClean
- [✓] ManualClean

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

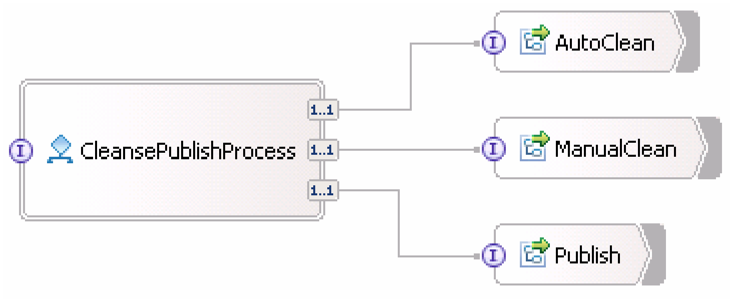


____ 8. With the partners wired to Imports, the type of binding must be specified. The binding indicates how the implementation should be reached. The implementation for the Imports will be provided by SCA components. For now, you can specify the SCA binding and later select the actual component implementation. Generate the binding to SCA

__ a. Right-click on **AutoClean** and select **Generate Binding….** Select **SCA Binding**

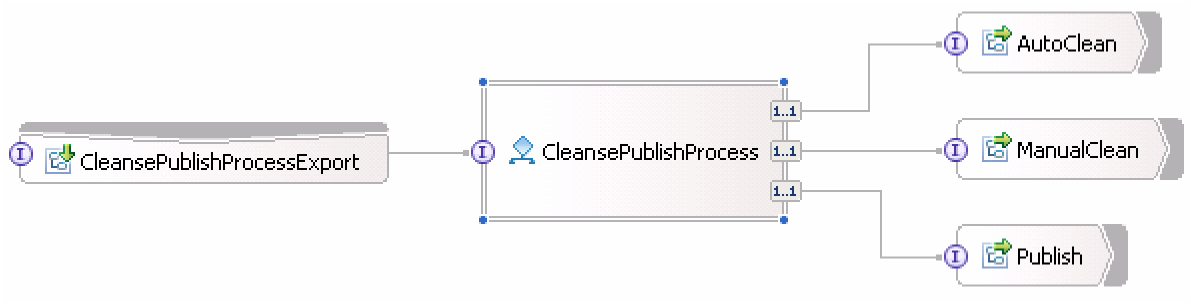
__ b. Right-click on **ManualClean** and select **Generate Binding….** Select **SCA Binding**

__ c. Right-click on **Publish** and select **Generate Binding….** Select **SCA Binding**



____ 9. 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 **Generate Export...** and click **SCA Binding**



__ b. Save the Assembly editor (**Ctrl +S**). The appropriate SCA component information will be generated for the CleansePublishProcess.

Part 4: Unit test CleansePublish business process

In this part of the lab, 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 before integrating with other components. The Integration Test Client will catch outgoing requests and wait until the response is completed and returned.

____ 1. Start the WebSphere Process Server test environment server

__ a. If using a remote testing environment, follow the directions provided in **Task: Adding remote server to WebSphere Integration Developer test environment** 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

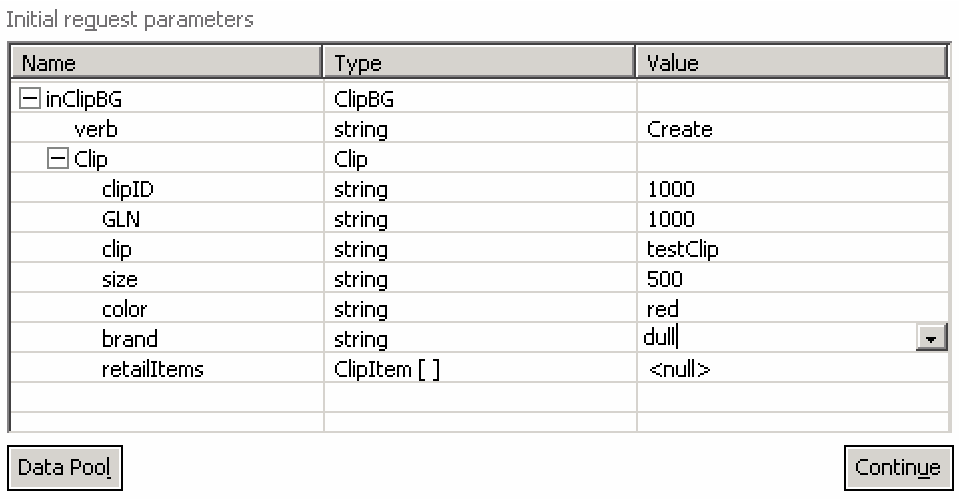
If using a local testing environment, change to the Servers view by selecting **Servers** tab

__ b. Right-click on **WebSphere Process Server V6.0** and select **Start** from the context menu

____ 2. After the server has started, in the Assembly editor, right-click on **CleansePublishBPEL** module and select **Test Component** from the context menu to test the business process component

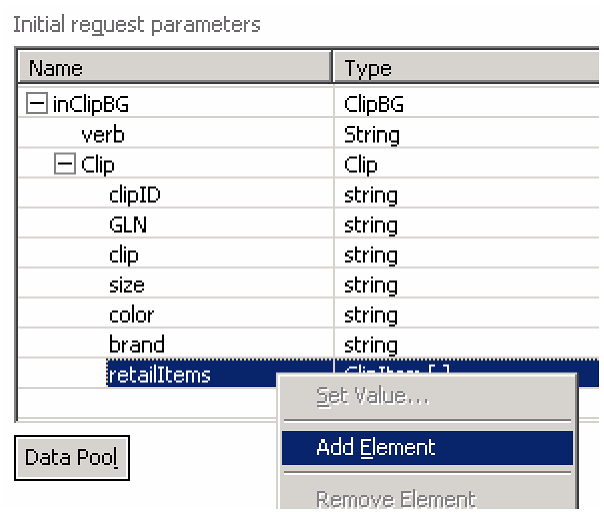
__ a. The Test Component editor will be opened with a table displaying the initial request parameter which is a business graph (BG) with verb for the business graph and the business object that is Clip. Enter the following values or of your choice:

- verb : **Create** (from the drop down list)
- clipID : **1000**
- GLN : **1000**
- clip : **testClip**
- size : **500**
- color : **red**
- brand : **dull**


Initial request parameters

Name	Type	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>

__ b. Right-click on the value for **retailItems** entry and select **Add Element**

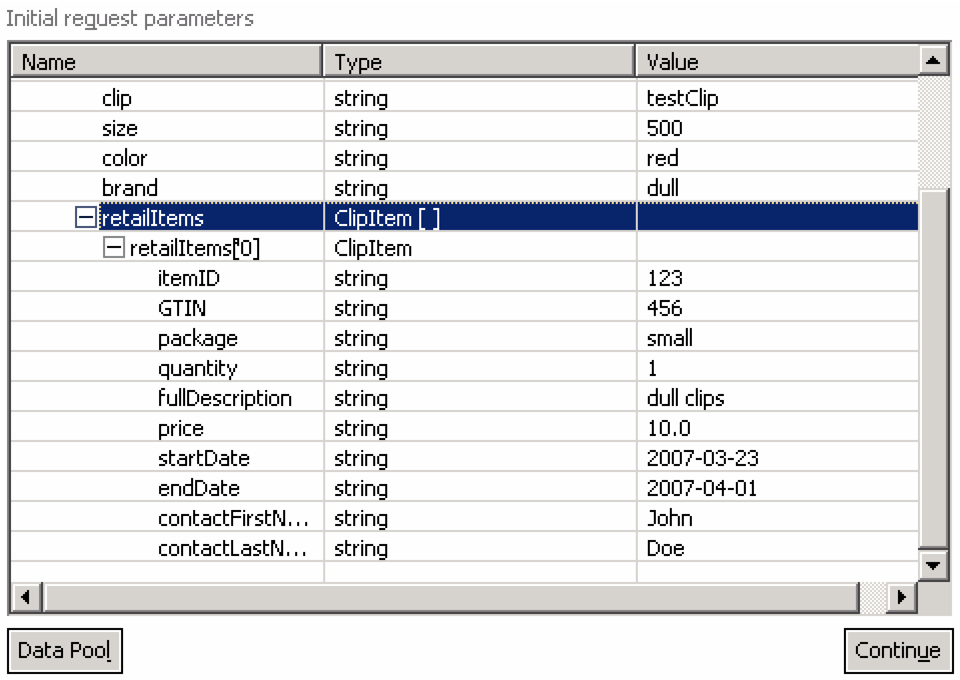

Initial request parameters

Name	Type
inClipBG	ClipBG
verb	String
Clip	Clip
clipID	string
GLN	string
clip	string
size	string
color	string
brand	string
retailItems	ClipItem [ ]

__ c. Enter **1** in the Add Element Dialog and click **OK**

__ d. Enter the following values or of your choice under **retailItems**:

- itemID : **123**
- GTIN : **456**
- package : **small**
- quantity : **1**
- fullDescription : **dull clips**
- price : **10.0**
- startDate : **2007-03-23**
- endDate : **2007-04-01**
- contactFirstName : **John**
- contactLastName : **Doe**



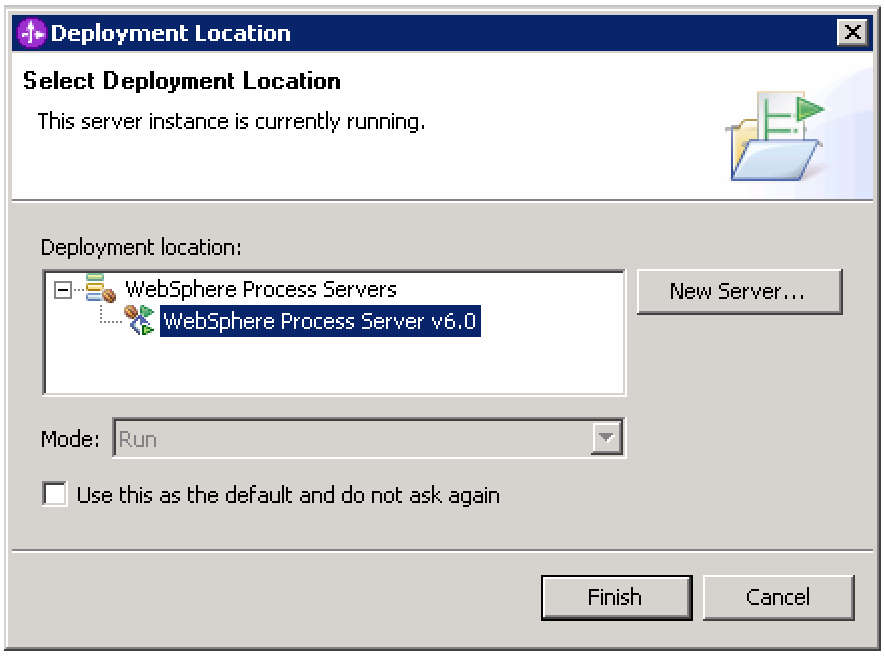
Initial request parameters

Name	Type	Value
clip	string	testClip
size	string	500
color	string	red
brand	string	dull
retailItems	ClipItem [ ]	
retailItems[0]	ClipItem	
itemID	string	123
GTIN	string	456
package	string	small
quantity	string	1
fullDescription	string	dull clips
price	string	10.0
startDate	string	2007-03-23
endDate	string	2007-04-01
contactFirstN...	string	John
contactLastN...	string	Doe

____ 3. Start the test

__ a. Click the **Continue** button under the list of initial request parameters

__ b. Select the **WebSphere Process Server V6.0** server in the **Choose a deployment location** dialog



Deployment Location

Select Deployment Location

This server instance is currently running.

Deployment location:

WebSphere Process Servers
WebSphere Process Server v6.0

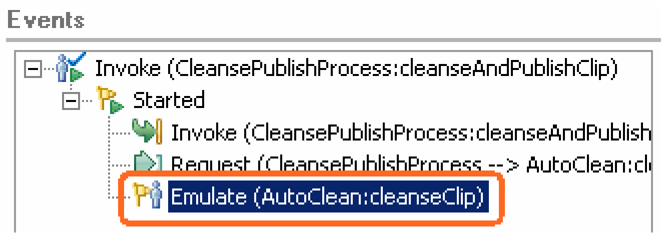
Mode: Run

Use this as the default and do not ask again

__ c. Click **Finish**. The CleansePublishBPELApp application will be published to the server and 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 responses 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


Events
Invoke (CleansePublishProcess:cleanseAndPublishClip)
Started
Invoke (CleansePublishProcess:cleanseAndPublish
Request (CleansePublishProcess --> AutoClean:cl
Emulate (AutoClean:cleanseClip)

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**.

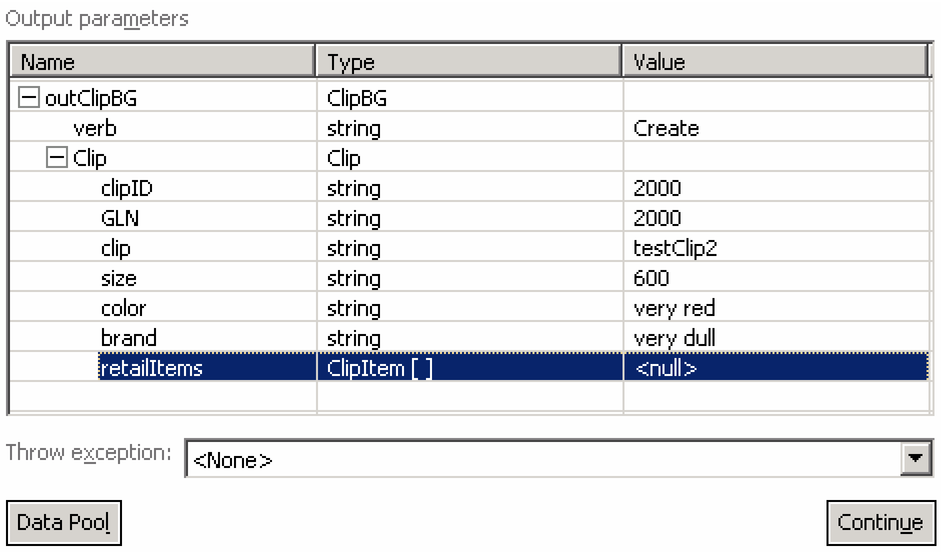
If using a remote testing environment, stop the server. Right click on WebSphere Process Server V6.0 server from the Servers view and select Stop from the context menu. Then follow the directions provided in **Task: Adding remote server to WebSphere Integration Developer test environment** to restart 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

__ c. There should also be a section for the Output parameters. Enter the following values or any values you choose:

- verb : **Create** (from the drop down list)
- clipID : 2**000**
- GLN : **2000**
- clip : **testClip2**
- size : **600**
- color : **very red**
- brand : **very dull**

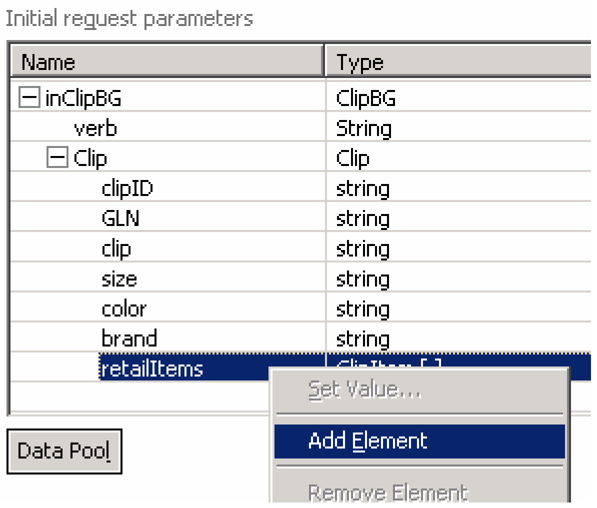


Output parameters

Name	Type	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>

Throw exception: <None>

__ d. Right-click on the value for **retailItems** entry and select **Add Element**



Initial request parameters

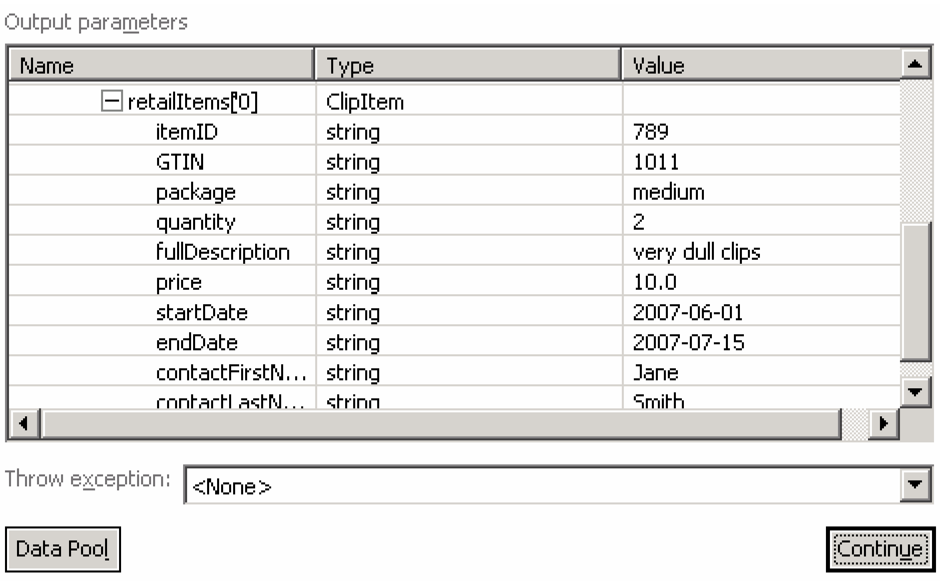
Name	Type
inClipBG	ClipBG
verb	String
Clip	Clip
clipID	string
GLN	string
clip	string
size	string
color	string
brand	string
retailItems	ClipItem [ ]

__ e. Enter **1** in the Add Element Dialog and click **OK**

__ f. Expand **retailItems** and enter the following values or any values you choose

- itemID : **789**
- GTIN : **1011**
- package : **medium**
- quantity : **2**
- fullDescription : **very dull clips**
- price : **10.0**
- startDate : **2007-06-01**
- endDate : **2007-07-15**

- contactFirstName : **Jane**
- contactLastName : **Smith**

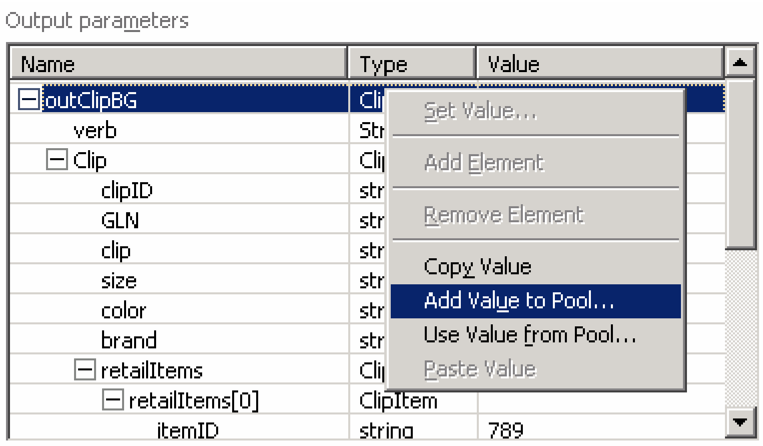


Output parameters

Name	Type	Value
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	2007-06-01
endDate	string	2007-07-15
contactFirstN...	string	Jane
contactLastN...	string	Smith

Throw exception: <None>

__ g. 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**



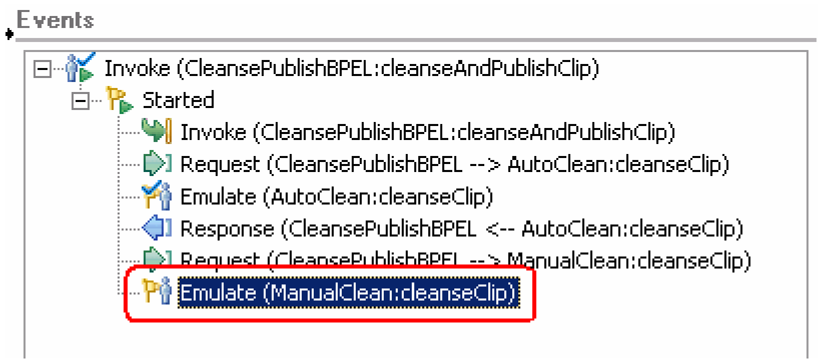
Output parameters

Name	Type	Value
outClipBG	Cli	
verb	Str	
Clip	Cli	
clipID	str	
GLN	str	
clip	str	
size	str	
color	str	
brand	str	
retailItems	Cli	
retailItems[0]	ClipItem	
itemID	string	789

__ h. Accept the default value name and click **OK**

__ i. Click the **Continue** button to send the response back to the business process engine

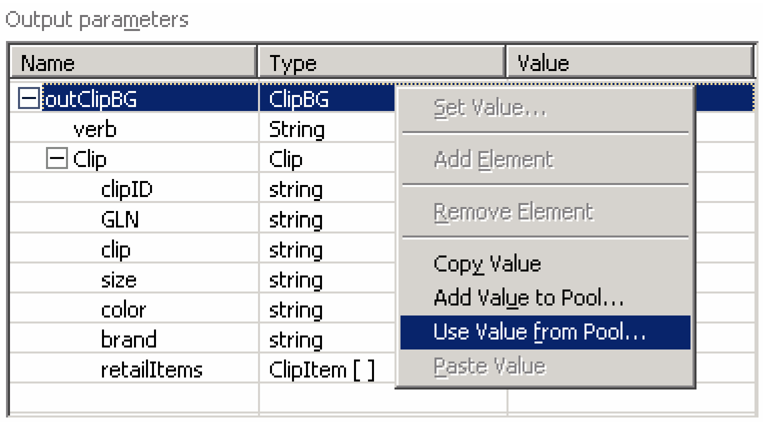
__ j. The response will be returned to the business process engine and processing will continue until the next invoke, ManualClean is reached. This activity also uses the cleanseClip interface. The test will wait until the response is specified. Notice the Input parameters are from the values you entered previously.



Events

- Invoke (CleansePublishBPEL:cleanseAndPublishClip)
  - Started
    - Invoke (CleansePublishBPEL:cleanseAndPublishClip)
    - Request (CleansePublishBPEL --> AutoClean:cleanseClip)
    - Emulate (AutoClean:cleanseClip)
    - Response (CleansePublishBPEL <-- AutoClean:cleanseClip)
    - Request (CleansePublishBPEL --> ManualClean:cleanseClip)
    - Emulate (ManualClean:cleanseClip)

__ k. Fill the Output parameters by selecting the **ClipBG** and selecting **Use Value from Pool…**



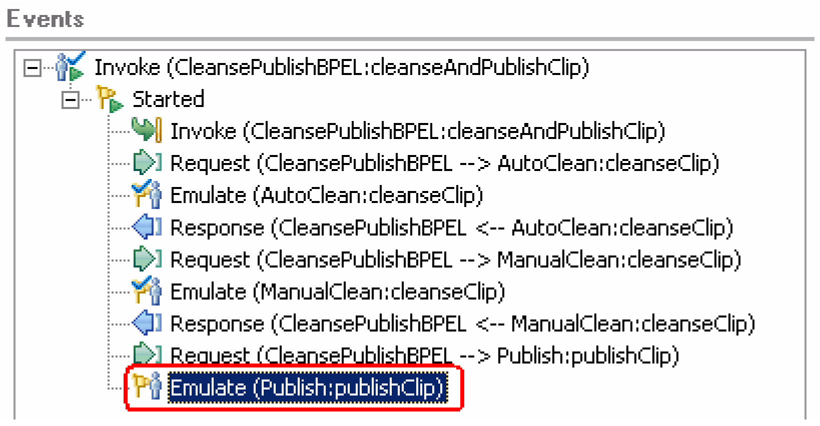
Output parameters

Name	Type	Value
outClipBG	ClipBG	
verb	String	
Clip	Clip	
clipID	string	
GLN	string	
clip	string	
size	string	
color	string	
brand	string	
retailItems	ClipItem [ ]	

__ l. Select **outClipBG** 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

__ m. Under outClipBG and Clip, change **clipID** and **GLN** to **3000**

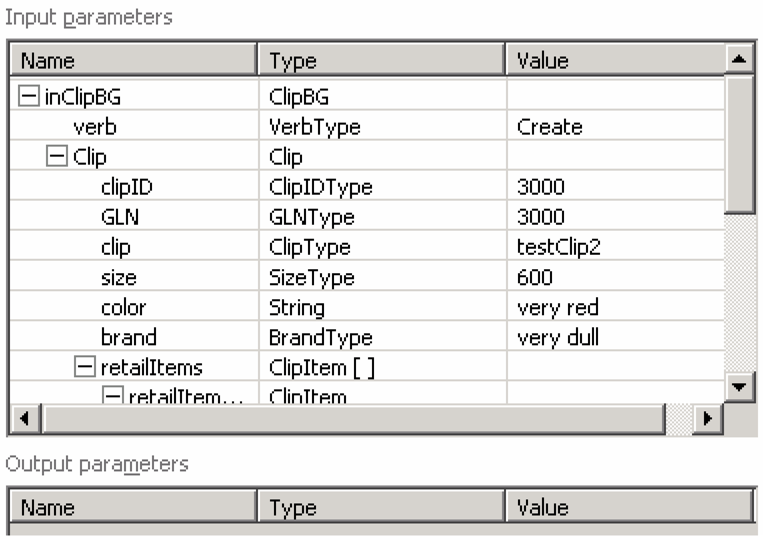
__ n. Click **Continue**. The response is returned and the business process will continue



Events

- Invoke (CleansePublishBPEL:cleanseAndPublishClip)
  - Started
    - Invoke (CleansePublishBPEL:cleanseAndPublishClip)
    - Request (CleansePublishBPEL --> AutoClean:cleanseClip)
    - Emulate (AutoClean:cleanseClip)
    - Response (CleansePublishBPEL <-- AutoClean:cleanseClip)
    - Request (CleansePublishBPEL --> ManualClean:cleanseClip)
    - Emulate (ManualClean:cleanseClip)
    - Response (CleansePublishBPEL <-- ManualClean:cleanseClip)
    - Request (CleansePublishBPEL --> Publish:publishClip)
    - Emulate (Publish:publishClip)

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



Name	Type	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 [ ]	
retailItem…	ClipItem	

Name	Type	Value

__ p. Click **Continue** to complete the request. The business process will complete and the test will end.

____ 5. Clean the WebSphere Process Server test environment

__ a. Right-click on the WebSphere Process Server V6 in the Servers view and select **Add and remove projects…** from the context menu

__ 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 Initialize the Workspace for a Lab Exercise, using the following values:

<WORKSPACE>

C:\Labfiles602\eXchange\CleanPublishBPEL\workspace

<PROJECT_INTERCHANGE>

C:\Labfiles602\eXchange\CleanPublishBPEL\solution\CleanPublishBPEL_PI.zip

<MODULE>

n/a

<DEPENDENT_LIBRARIES>

n/a

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

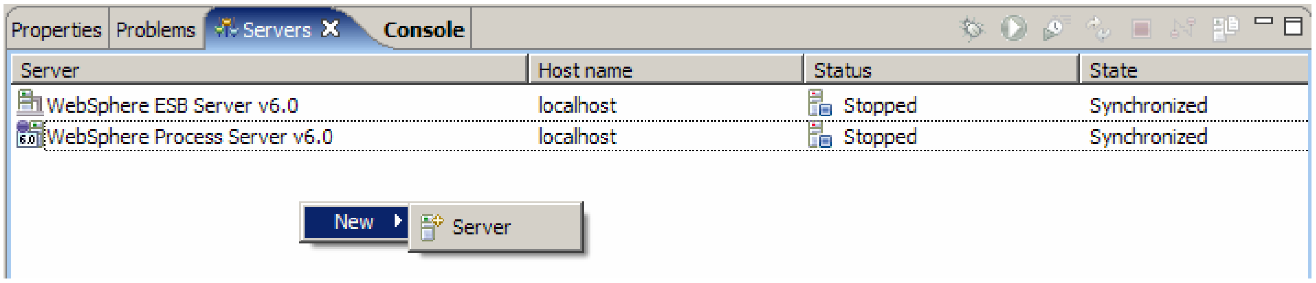
Task: Adding remote server to WebSphere Integration Developer test environment

This task describes how to add a remote server to the WebSphere Integration Developer test environment. The sample you will use is 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.

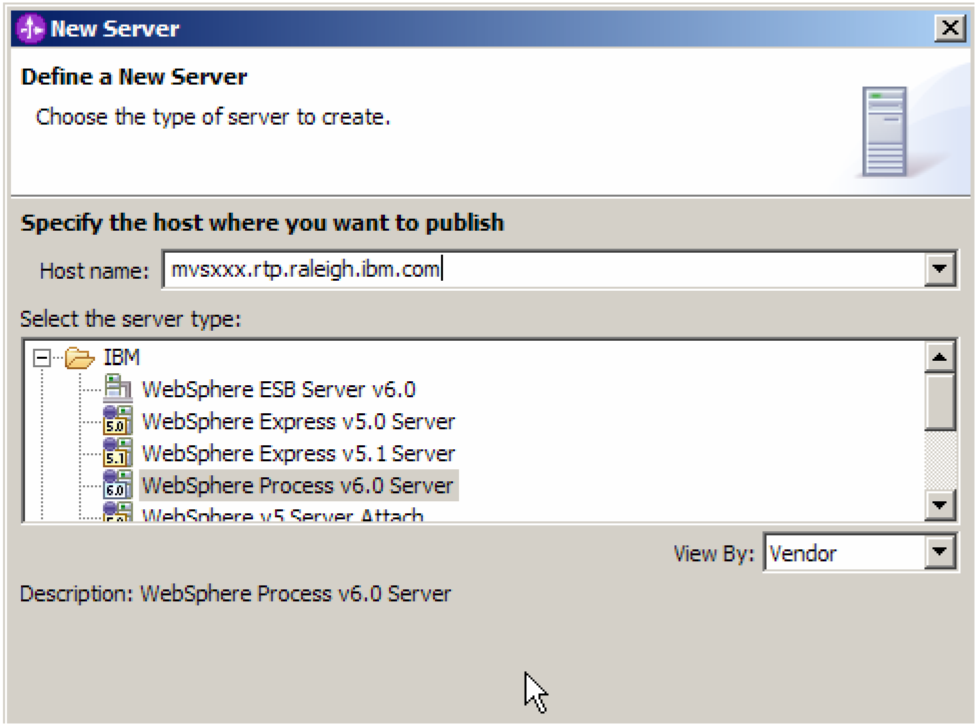
____ 2. Select **New → Server.**



Server	Host name	Status	State
WebSphere ESB Server v6.0	localhost	Stopped	Synchronized
WebSphere Process Server v6.0	localhost	Stopped	Synchronized

____ 3. Specify host name 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:

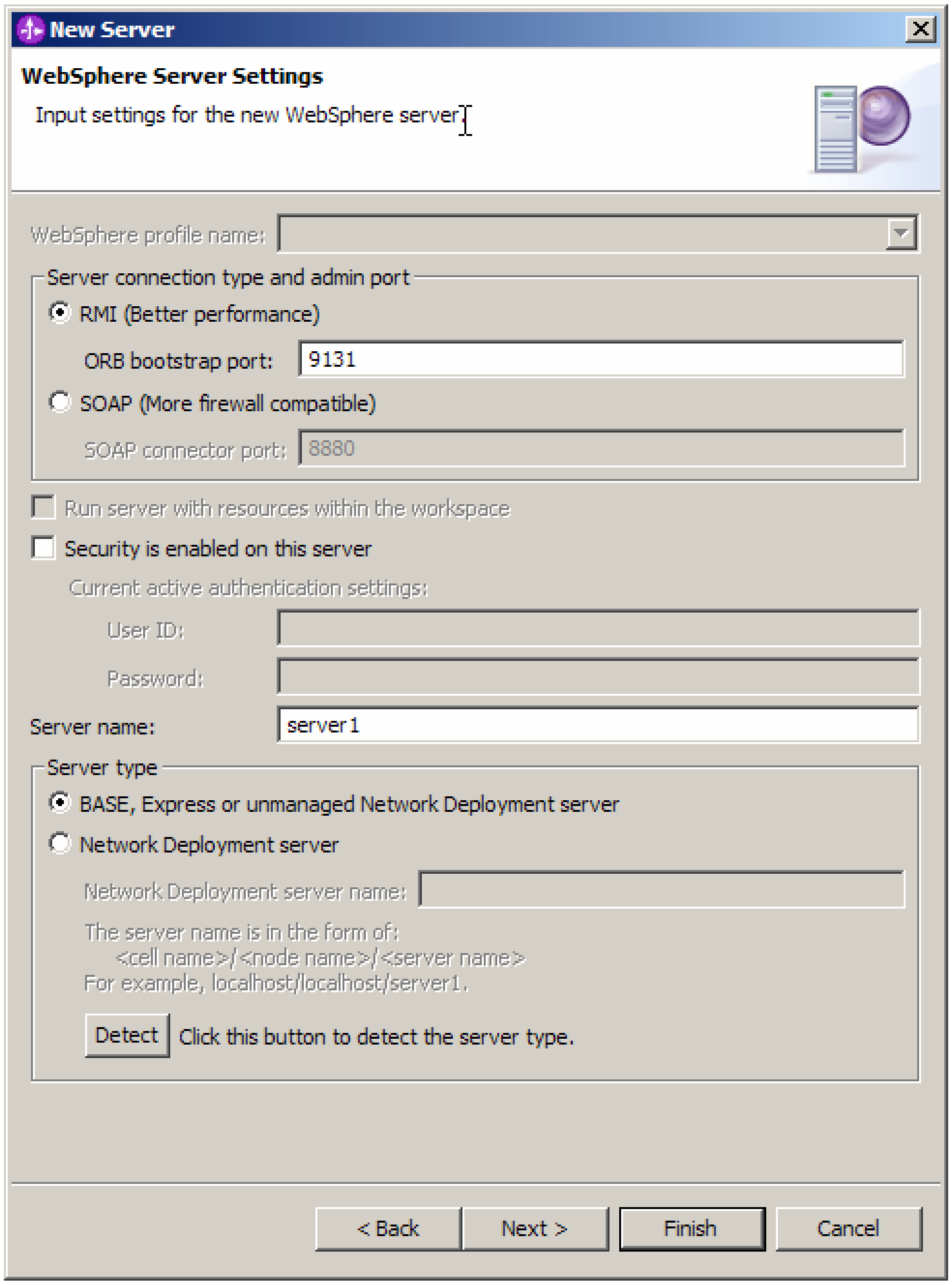
- IBM
  - WebSphere ESB Server v6.0
  - WebSphere Express v5.0 Server
  - WebSphere Express v5.1 Server
  - WebSphere Process v6.0 Server
  - WebSphere v5 Server Attach

View By: Vendor

Description: WebSphere Process v6.0 Server

____ 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:

Server connection type and admin port

[✓ RMI (Better performance)]

ORB bootstrap port: 9131

SOAP (More firewall compatible)

SOAP connector port: 8880

Run server with resources within the workspace

Security is enabled on this server

Current active authentication settings:

User ID:

Password:

Server name: server1

Server type

[✓ BASE, Express or unmanaged Network Deployment server]

Network Deployment server

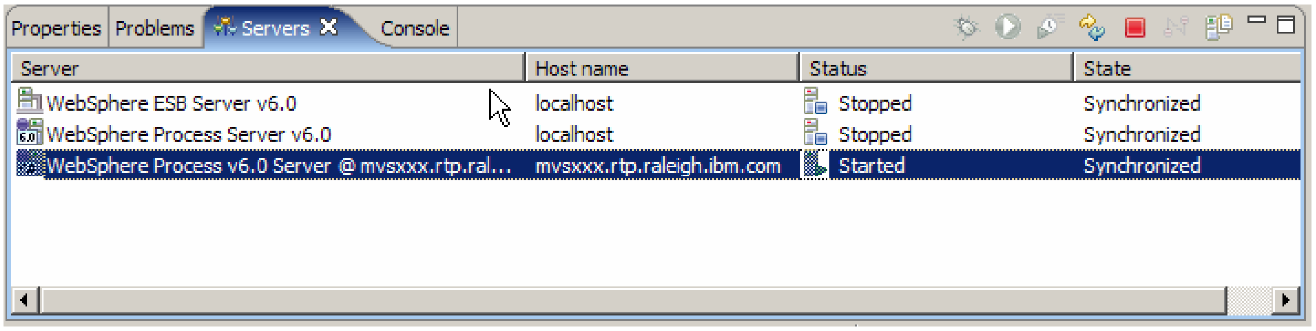
Network Deployment server name:

The server name is in the form of:
<cell name>/<node name>/<server name>
For example, localhost/localhost/server1.

Detect Click this button to detect the server type.

____ 7. Click **Finish**.

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



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

____ 9. Start the remote server if it is not already started. WebSphere Integration Developer does not support starting remote servers from the Server View.

____ 10. From a command prompt, telnet to the remote system if needed:

‘**telnet <HOSTNAME> <TELNET_PORT>**’

userid : **<USERID>**

pw : **<PASSWORD>**

____ 11. Navigate to the bin directory for the profile being used:

cd <WAS_HOME>/profiles/<PROFILE_NAME>/bin

____ 12. Run the command file to start the server: **./startServer.sh <SERVER_NAME>**

____ 13. Wait for status message indicating server has started:

```
ADMU3200I: Server launched. Waiting for initialization status.

ADMU3000I: Server cl1sr01 open for e-business; process id is 0000012000000002
```