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 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)

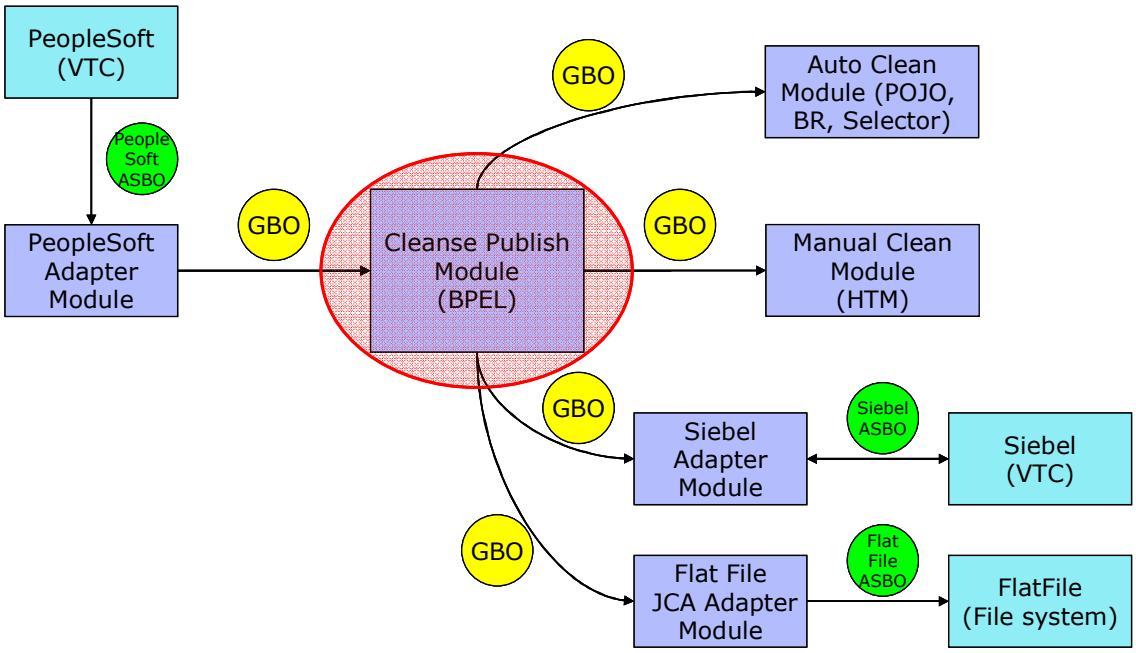
What you should be able to do

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

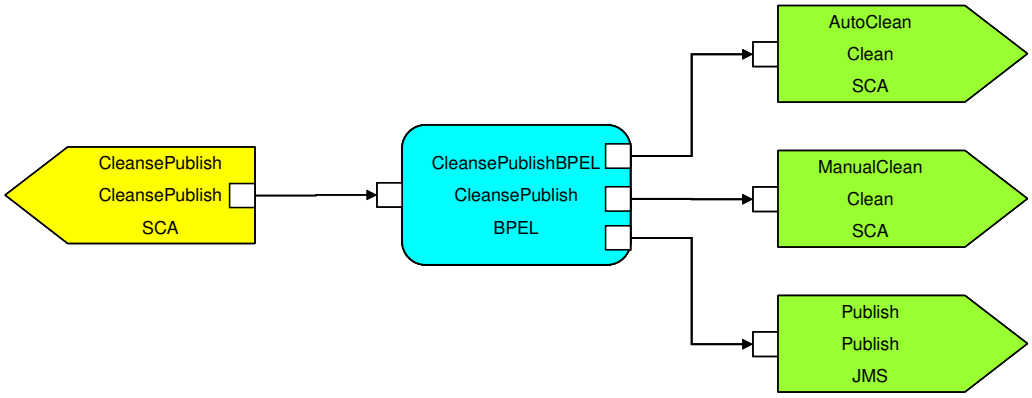
Introduction

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™ 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>	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:\Labfiles60	/tmp/Labfiles60
<WORKSPACE>	C:\Labfiles60\eXchange\CleanPublishBPEL\workspace	/tmp/Labfiles60/eXchange/CleanPublishBPEL/workspace
<TEMP>	C:\temp	/tmp
<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® 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	
<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

____ 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

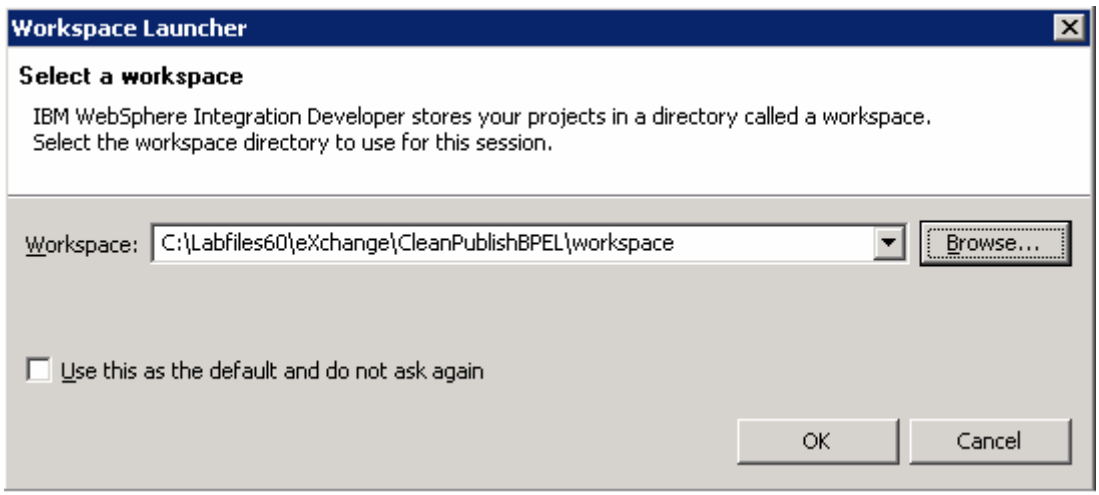
<DEPENDENT_LIBRARIES>

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.**



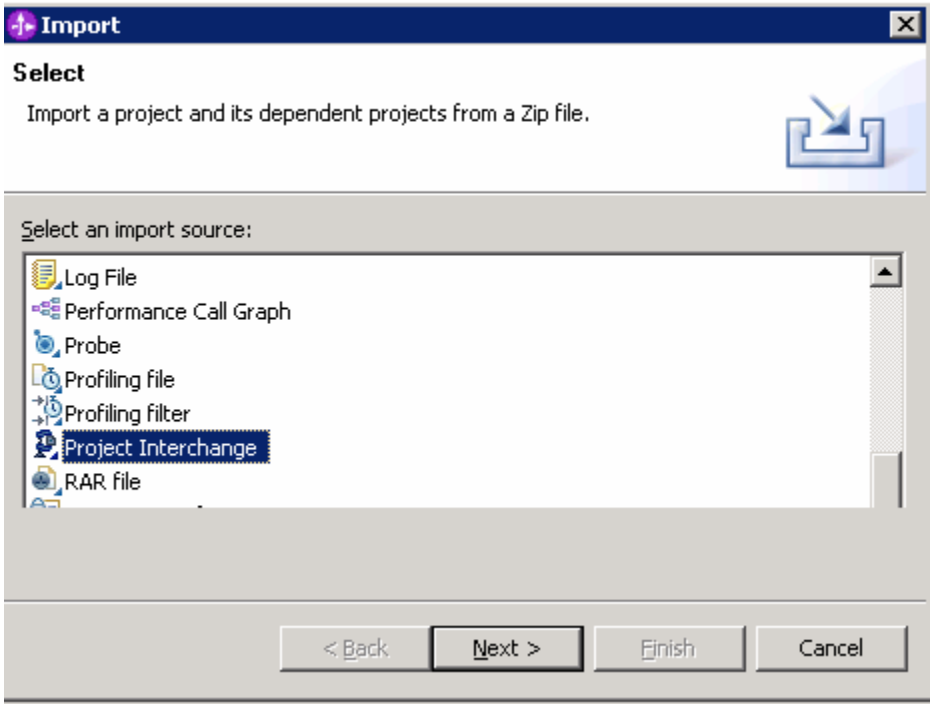
__ c. When WebSphere Integration Developer V6.0 opens, close the **Welcome page** by clicking on the Go to the workbench icon (bent over arrow at top-right).

__ 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…**

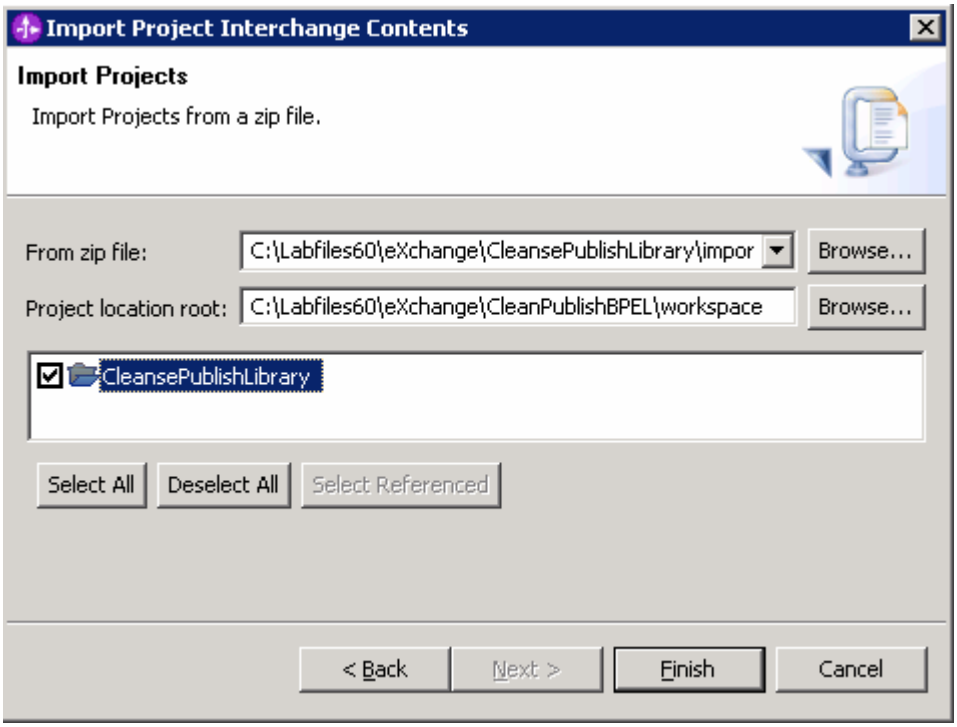
__ b. In the Import dialog, scroll down and select **Project Interchange.**



__ c. Click **Next.**

__ d. In the Import Projects dialog, initialize the From zip file: field to <**PROJECT_INTERCHANGE>.**

__ e. Click the **Select All** button.

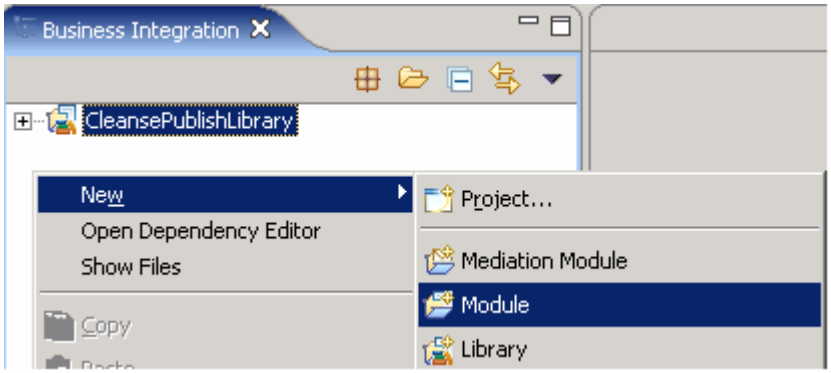


__ f. Click **Finish.**

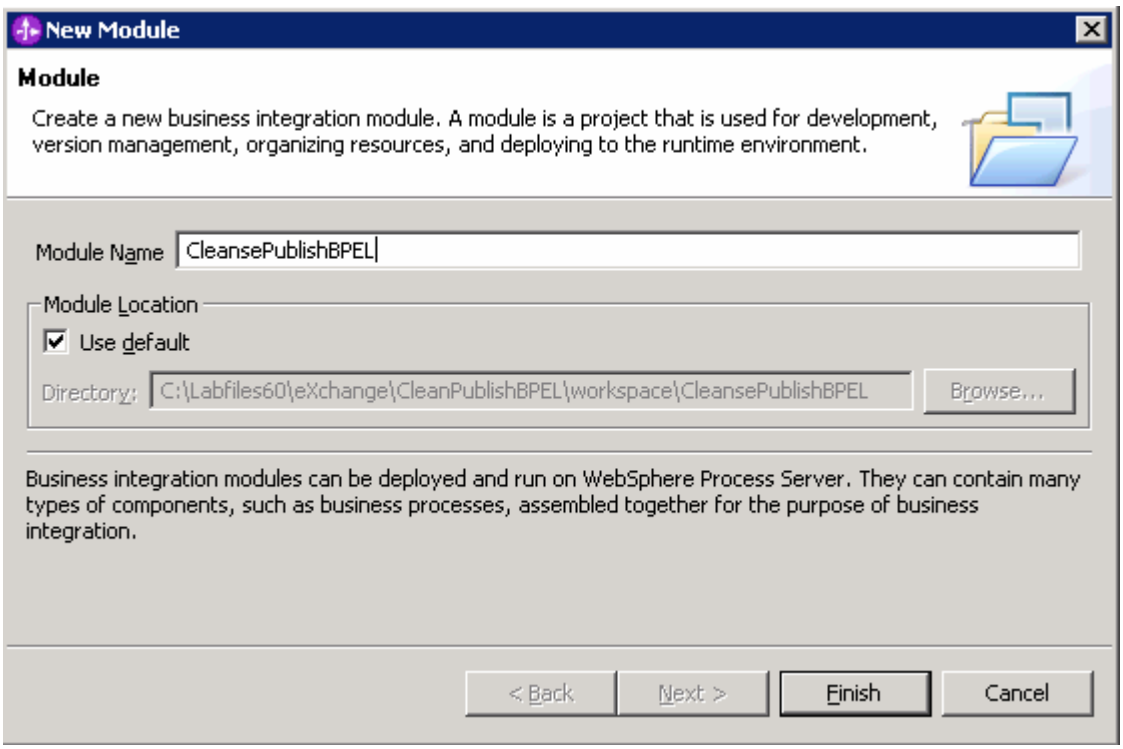
____ 4. If this lab requires that you create a Business Integration module called **<MODULE>**, complete these steps:

__ a. Right click on the background of the Business Integration view to access the pop-up menu.

__ b. Select **New > Module.**



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

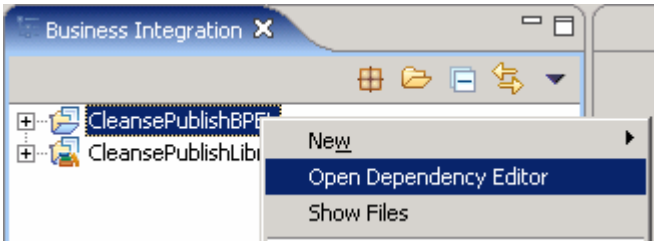


__ d. Click **Finish.**

____ 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.

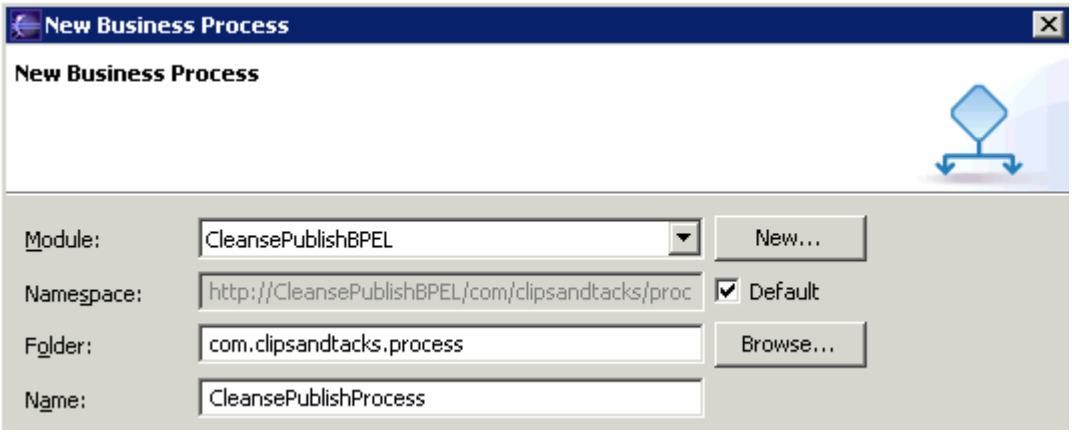
____ 1. Create an empty business process named CleansePublishBPEL.

__ a. Expand **CleansePublishBPEL** and **Business Logic i**n 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.



__ e. Click **Next**.

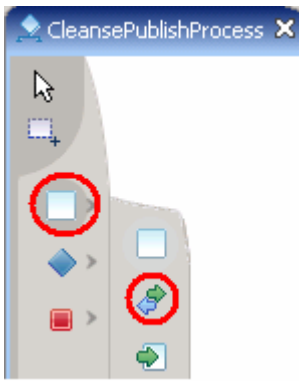
__ f. Select the radio button for **Select an Existing Interface**.

__ 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.

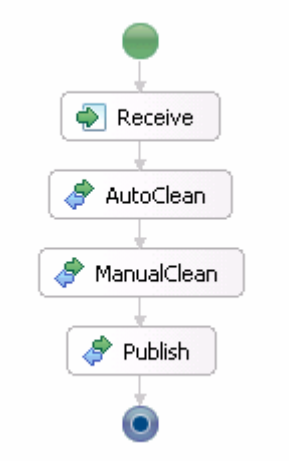
____ 2. The CleansePublishProcess is a simple business process composed of a series of steps that form the business logic. Define the business logic with Invoke activities.

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



__ 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.

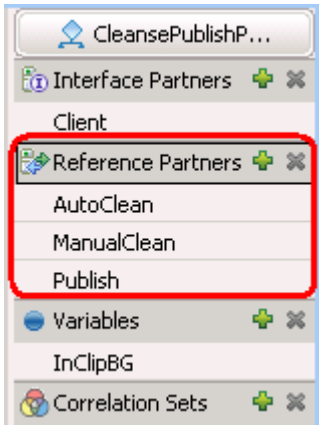
__ f. Select the **Properties** view and the **Description** Tab.

__ 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.

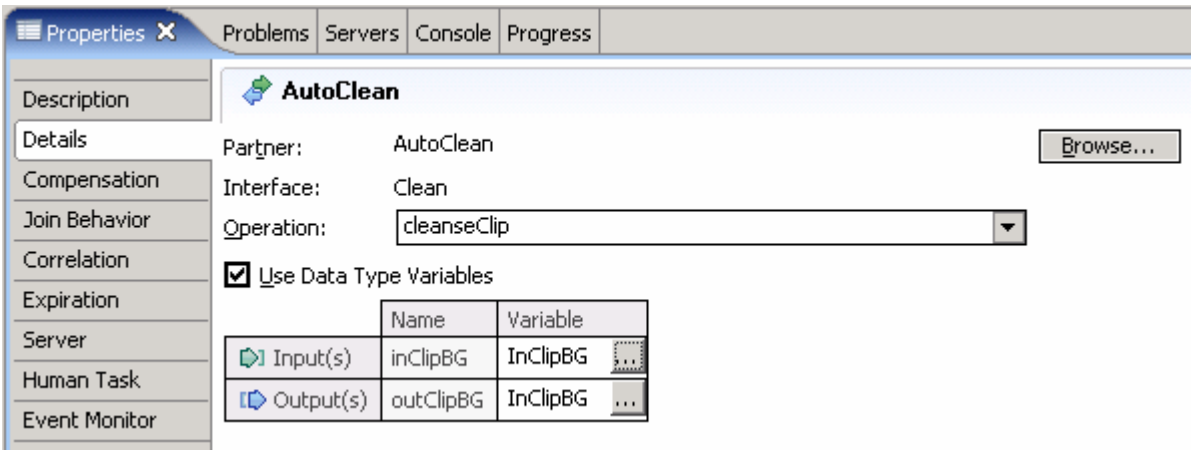
__ e. Select **InClipBG** in the Select Variable for inclipBG window.

__ f. Click **OK**.

__ g. Click the '**…**' button for the **Output** variable.

__ h. Click **InClipBG** in the Select Variable for outClipBG window.

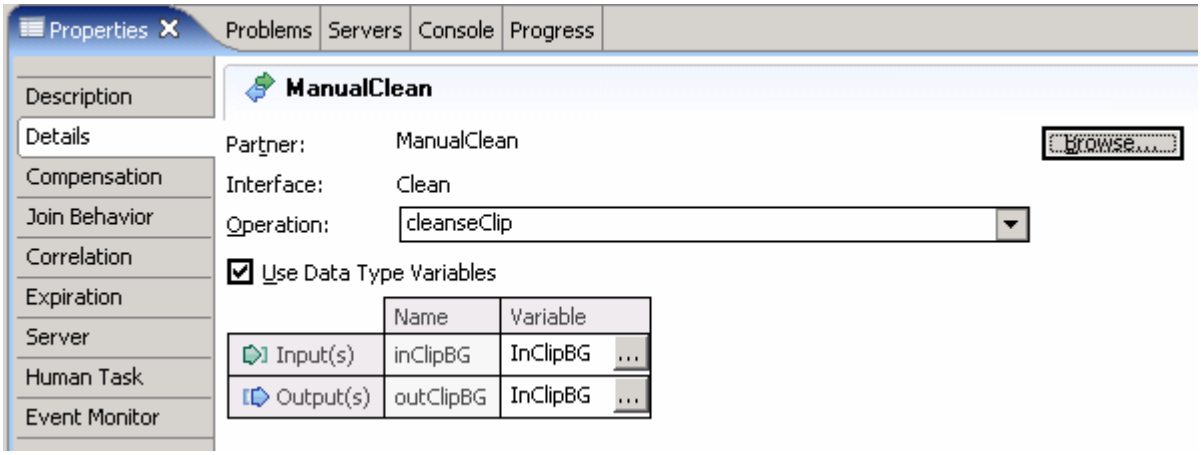
__ i. Click **OK**.



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.

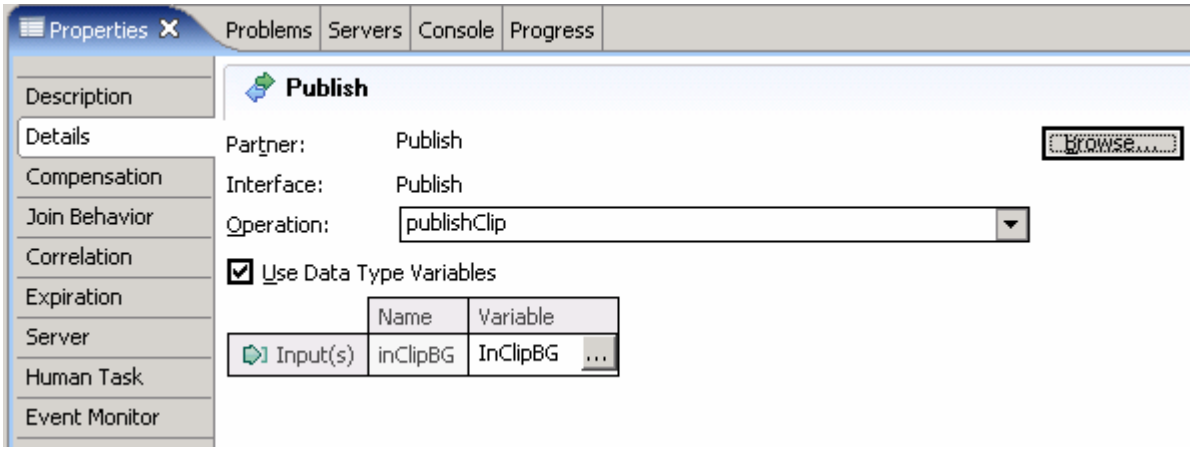
__ j. Select the **ManualClean** Invoke activity in the Assembly editor and then select Properties View.

__ k. Set the Partner to **ManualClean** and specify InClipBG for the Input and Output variables.



__ l. 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.



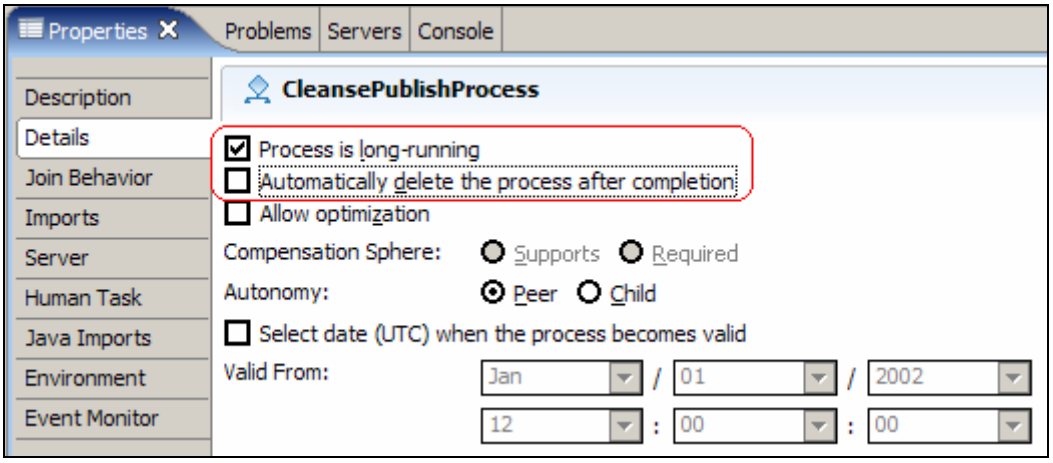
____ 5. Business processes can be short-running or long-running (interruptible). Short-running processes run in a single transaction and are very fast whereas long-running business processes are executed over multiple transactions and can resume after halting and maintain state. In CleansePublishProcess, 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**.



__ 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.

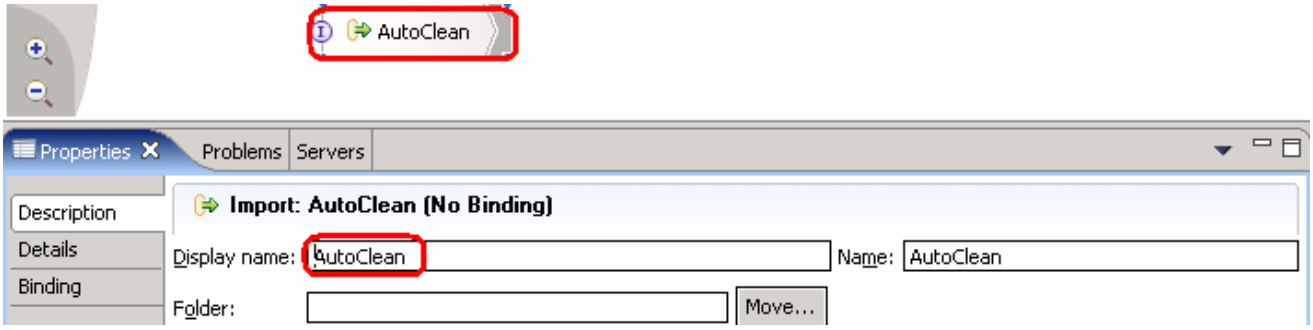
__ a. Expand **CleansePublishLibrary** and **Interfaces** in the Business Integration view.

__ 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.



__ 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.

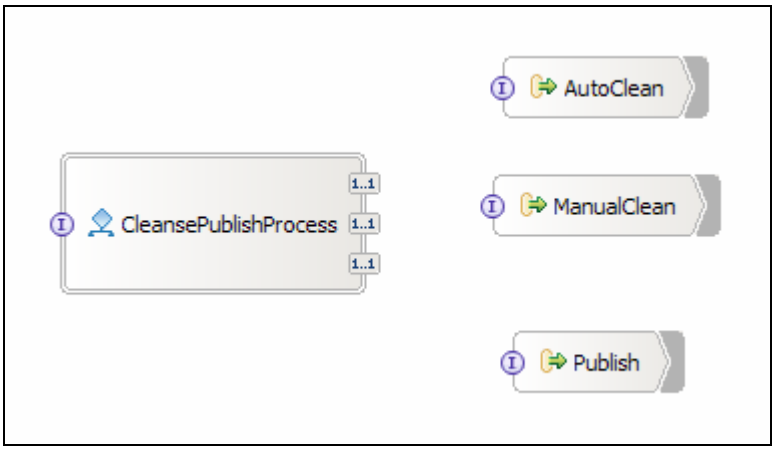
__ g. Select **Import with No Binding** in the Component Creation window and click **OK**.

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

__ i. Drop **Publish** onto the Assembly Editor.

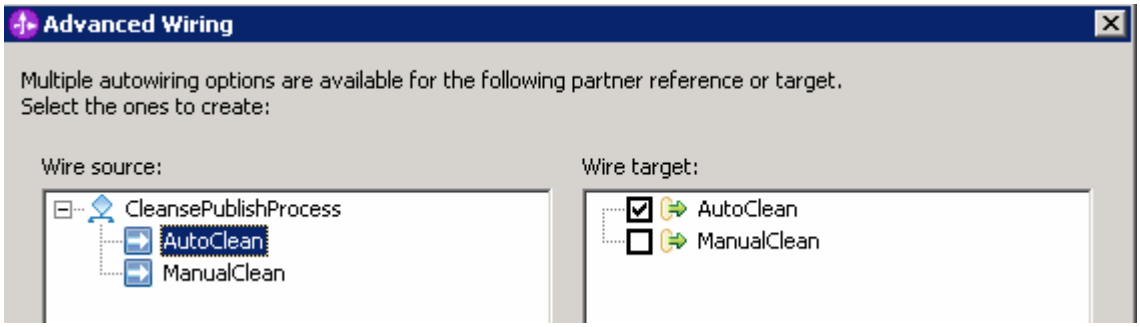
__ j. Select **Import with No Binding** in the Component Creation window and click **OK**.

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

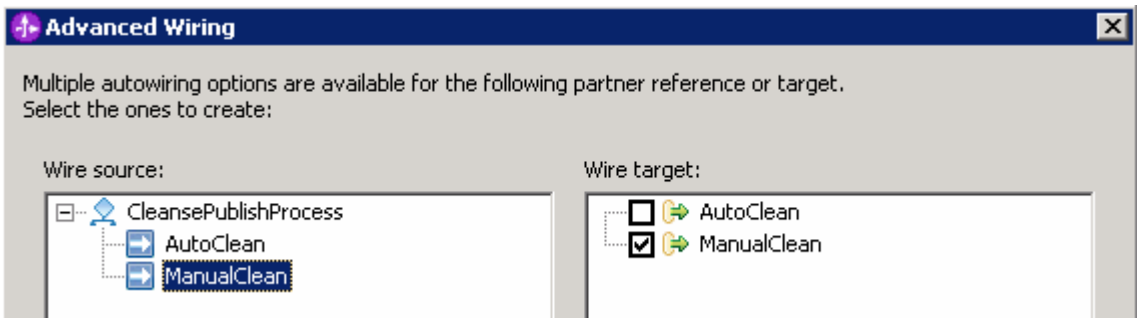


__ l. Right-click on the CleansePublishProcess and select **Wire to Existing**. The editor will attempt to wire those partner references to the imports automatically.

__ 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.



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



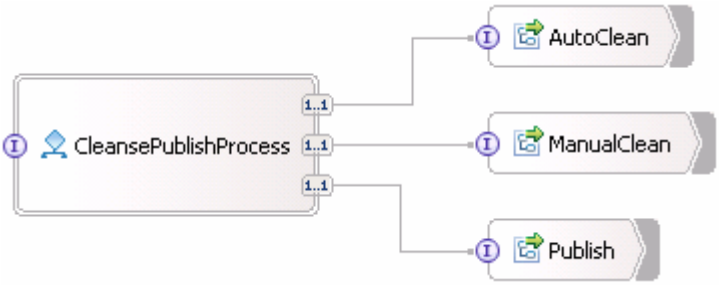
__ o. Click **OK**. The Publish partner is wired directly to the Publish import.

____ 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**.

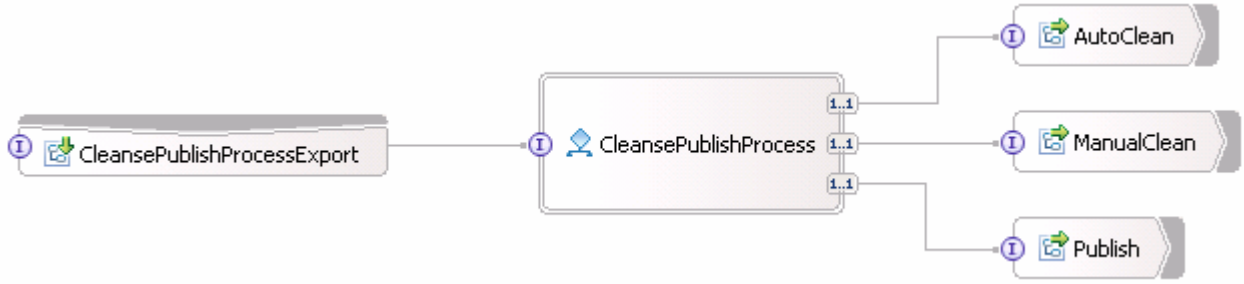
__ b. Right-click on **ManualClean** and select **Generate Binding…**. Select **SCA Binding**.

__ 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**.



__ 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.

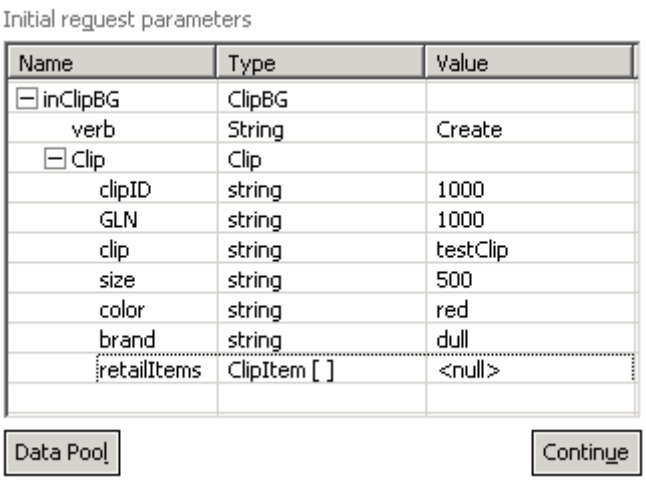
__ a. If using a remote testing environment, follow the directions provided in **Task: Adding Remote Server to WebSphere Integration Developer Test Environment** at the end of this document to add a server to the WebSphere Integration Developer test environment and start it. This is especially true for 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.

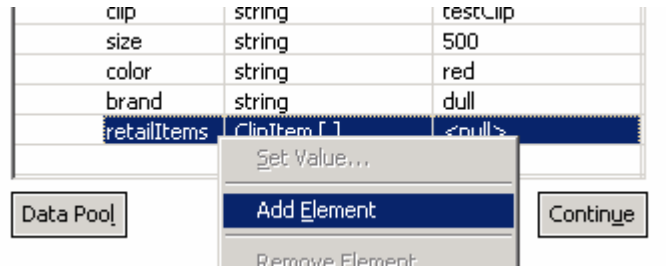
__ b. Right-click on **WPS Server v6.0**. Select **Start**.

____ 2. After the server has restarted, in the Assembly editor, right-click on **CleansePublishProcess** and select **Test Component** to test the business process component.

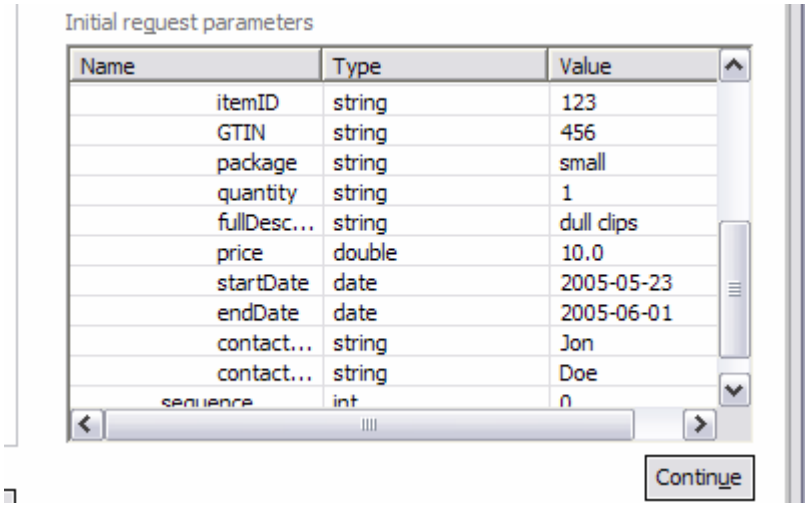
__ a. A test instance will be opened with a table displaying the initial request parameter which is a business graph (BG). There is a verb for the business graph and the business object, Clip. Fill in the values for the verb and the business object.



__ b. For the **retailItems** value, right-click on the value and select **Add Element**.



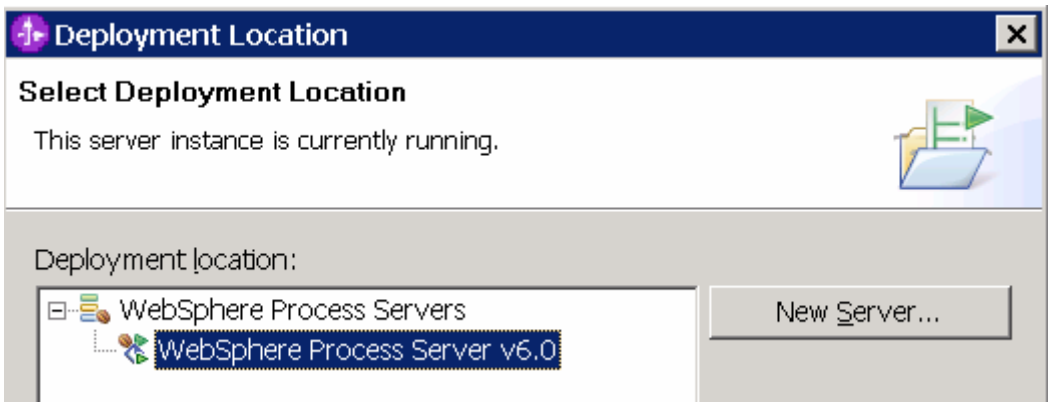
__ c. Scroll down and fill in the vales according to the graphic or any values you choose. Again remember to specify values which are compatible with the value type.



____ 3. Start the test.

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

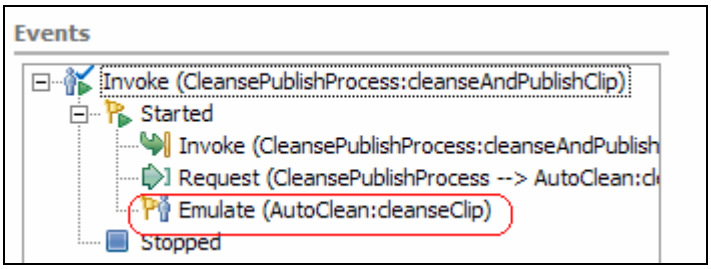
__ b. Expand **WebSphere Process Server** in the Deployment Location window and select **WebSphere Process Server v6.0**.



__ 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 **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.

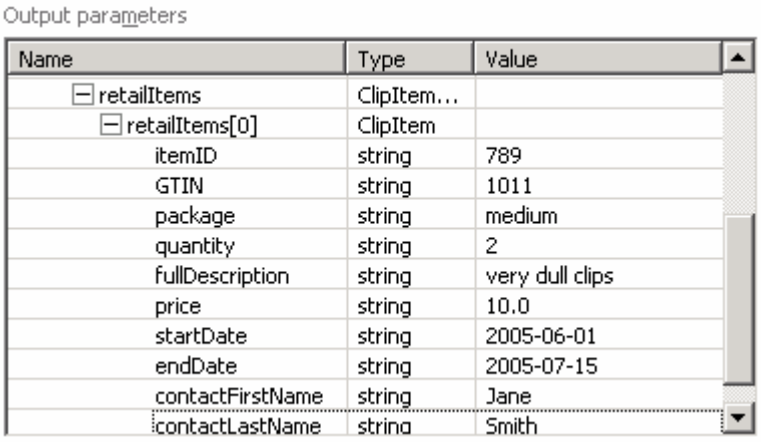
__ b. Under the Detailed Properties section, you should see the Input parameters that were sent as the request to the service. The parameters should match the values you specified when starting the test. There should also be a section for the Output parameters. Enter the following values or any values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

Output parameters

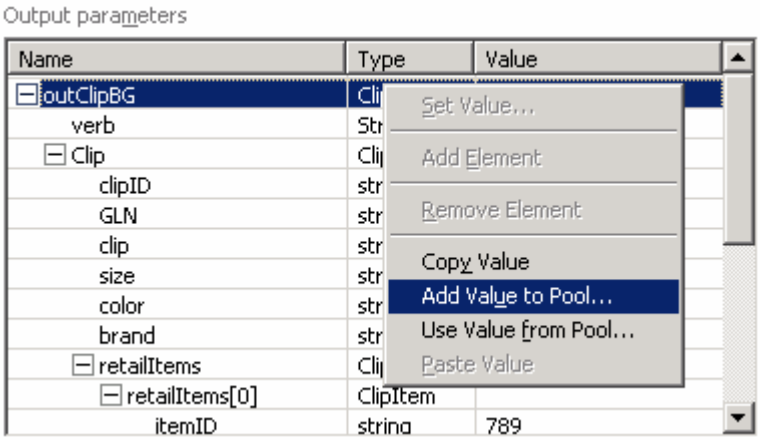
Name	Type	Value
outClipBG	ClipBG	
verb	String	Create
Clip	Clip	
clipID	string	2000
GLN	string	2000
clip	string	testClip2
size	string	600
color	string	very red
brand	string	very dull
retailItems	ClipItem [ ]	<null>

__ 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.



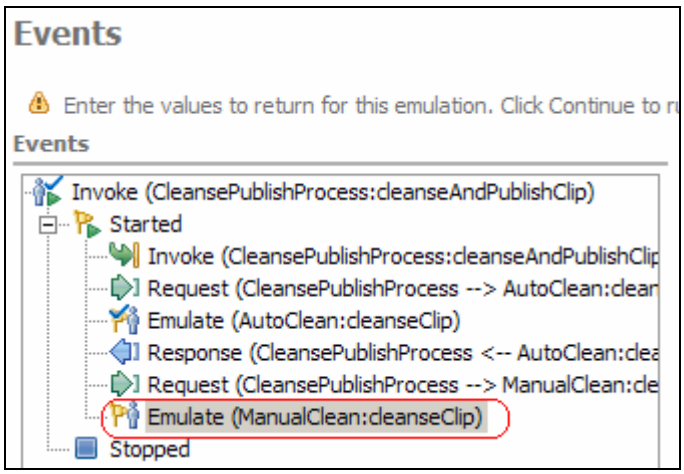
__ e. Before continuing the test, save the parameters you just entered into the data pool for reuse. Scroll to the top of the Output parameters and right-click on **ClipBG** and select **Add Value to Pool**.



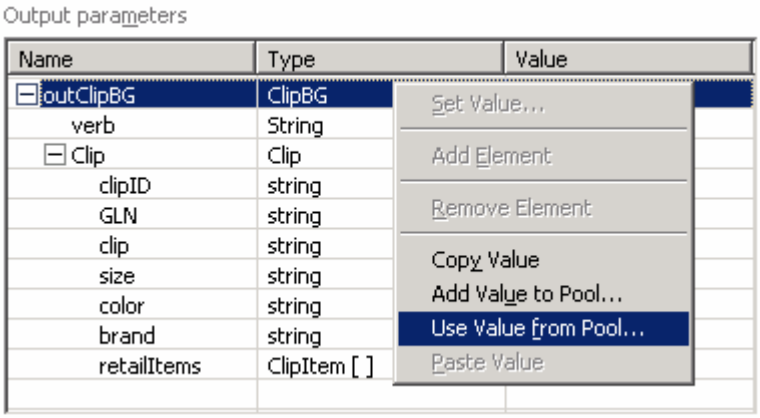
__ f. Accept the default value name and click **OK**.

__ 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.



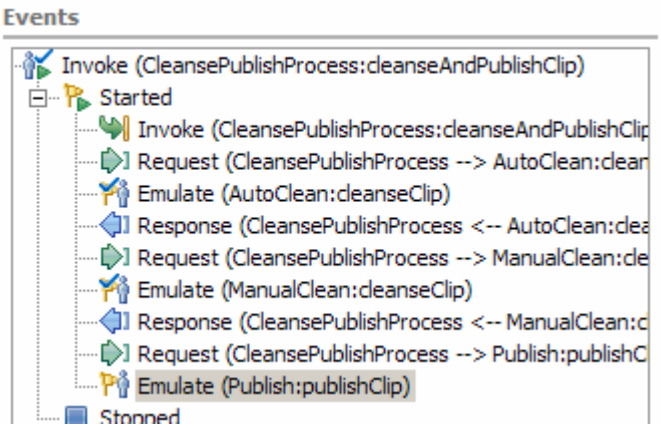
__ i. Fill out the Output parameters by selecting the **ClipBG** and selecting **Use Value from Pool…**



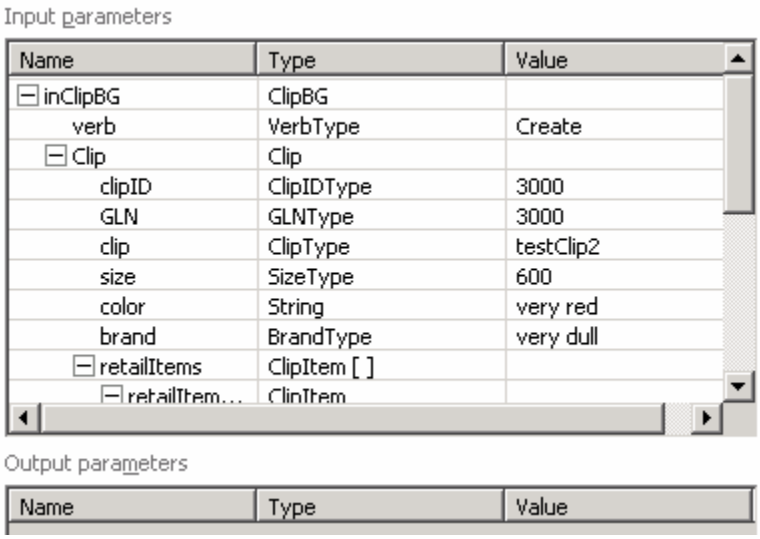
__ j. Select **inClipBG** from the list of values and click **OK**. The Output parameters will be populated with the values from the pool and can be easily reused or changed.

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

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



__ 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.



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

<WORKSPACE>

C:\Labfiles60\eXchange\CleanPublishBPEL\workspace

<PROJECT_INTERCHANGE>

C:\Labfiles60\eXchange\CleanPublishBPEL\solution\CleanPublishBPEL_PI.zip

<MODULE>

n/a

<DEPENDENT_LIBRARIES>

n/a

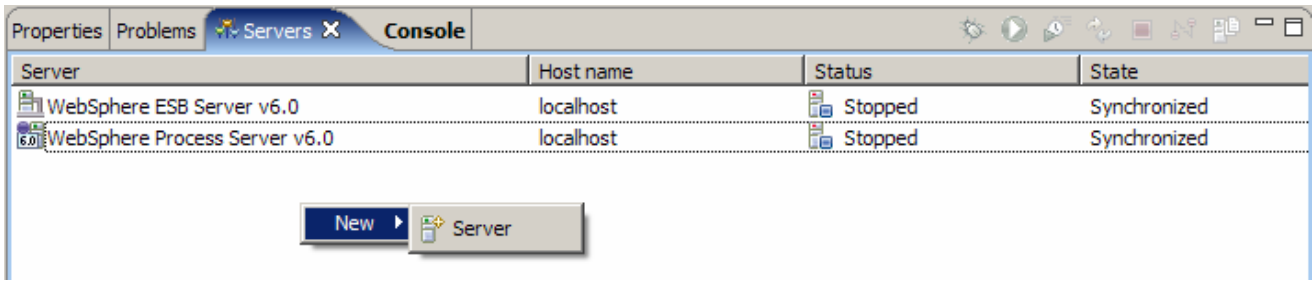
____ 2. Continue with **Part 4: Unit Test CleansePublish Business Process**.

Task: Adding Remote Server to WebSphere Integration Developer Test Environment

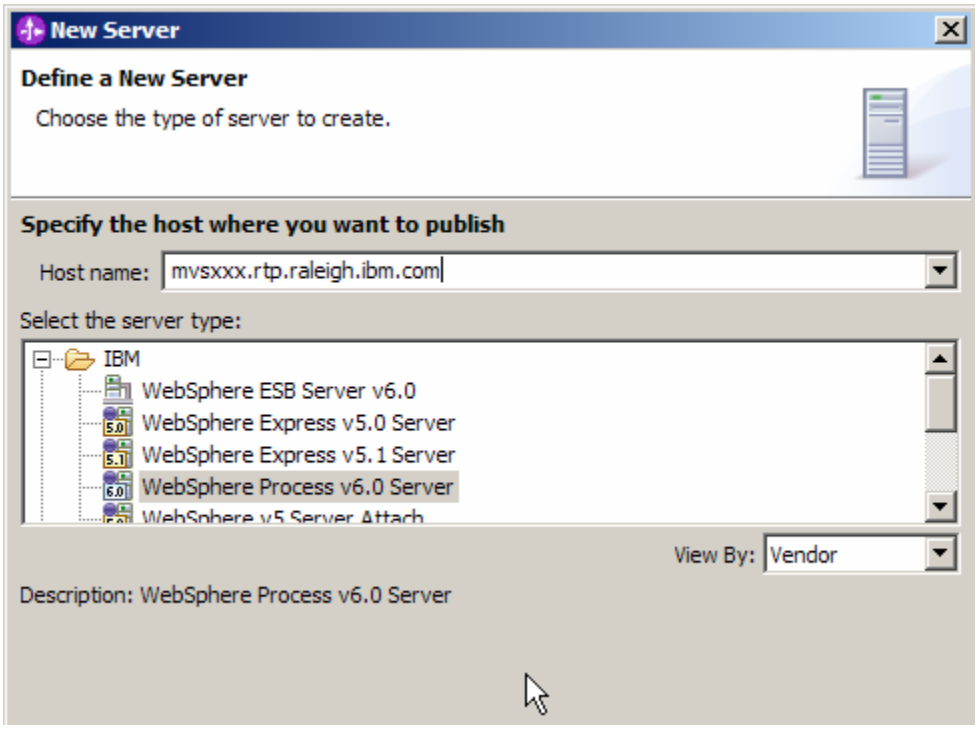
This task describes how to add a remote server to the WebSphere Integration Developer Test environment. This sample will use a z/OS machine.

Create a new remote server.

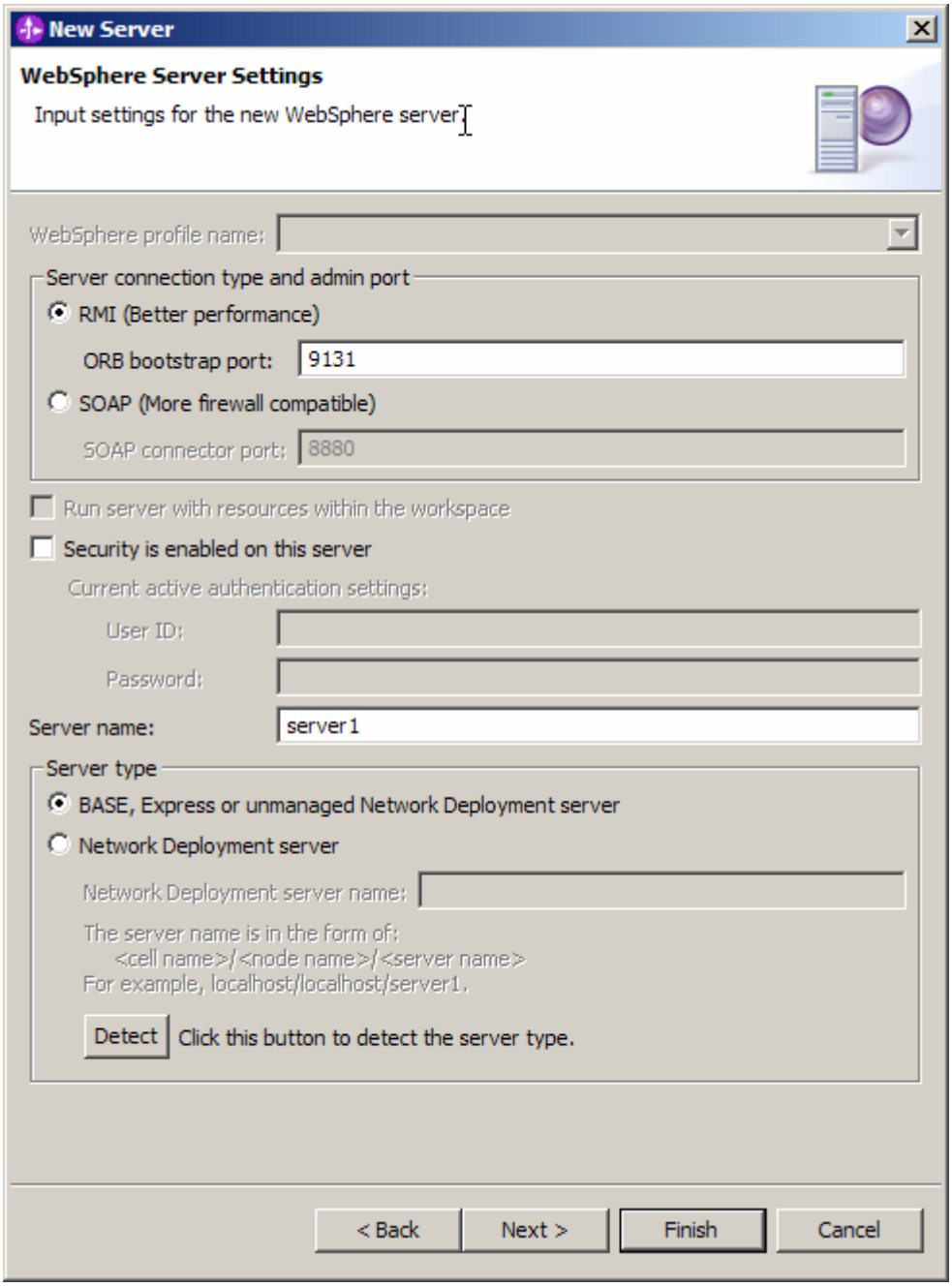
1. Right click on the background of the Servers view to access the pop-up menu.
2. Select **New > Server.**



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

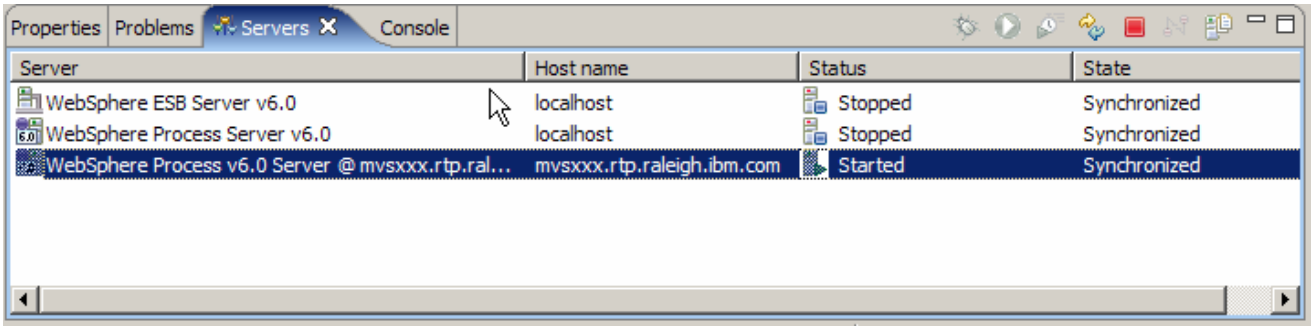


5. Click **Next.**
6. On the WebSphere Server Settings page, select the radio button for **RMI** and change the ORB bootstrap port to the correct setting (**<BOOTSTRAP_PORT>**).



7. Click **Finish**.

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



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