

## AutoClean – Business rules and selectors

What this exercise is about .....	2
Lab requirements .....	2
What you should be able to do .....	2
Introduction .....	3
Description of the module .....	3
Exercise instructions .....	4
Part 1: Initialize the workspace for this lab exercise .....	6
Part 2: Create a module and import Java implementation.....	9
Part 3: Create a business rule group .....	15
Part 4: Create a business rule Set .....	17
Part 5: Create a selector .....	24
Part 6: Create an export on assembly editor .....	27
Part 7: Publish the AutoCleanApp .....	29
Update business rules using the business rule editor Web tool .....	30
Part 8: Test the business rule using component test.....	35
What you did in this exercise .....	38
Solution instructions .....	39
Task: Adding remote server to WebSphere Integration Developer test environment .....	40

---

## What this exercise is about

This exercise is about learning the basics of Business Rules and Selectors while learning how to use WebSphere Integration Developer V6.0.2 and the WebSphere Process Server V6.

## Lab requirements

List of system and software required for the student to complete the lab:

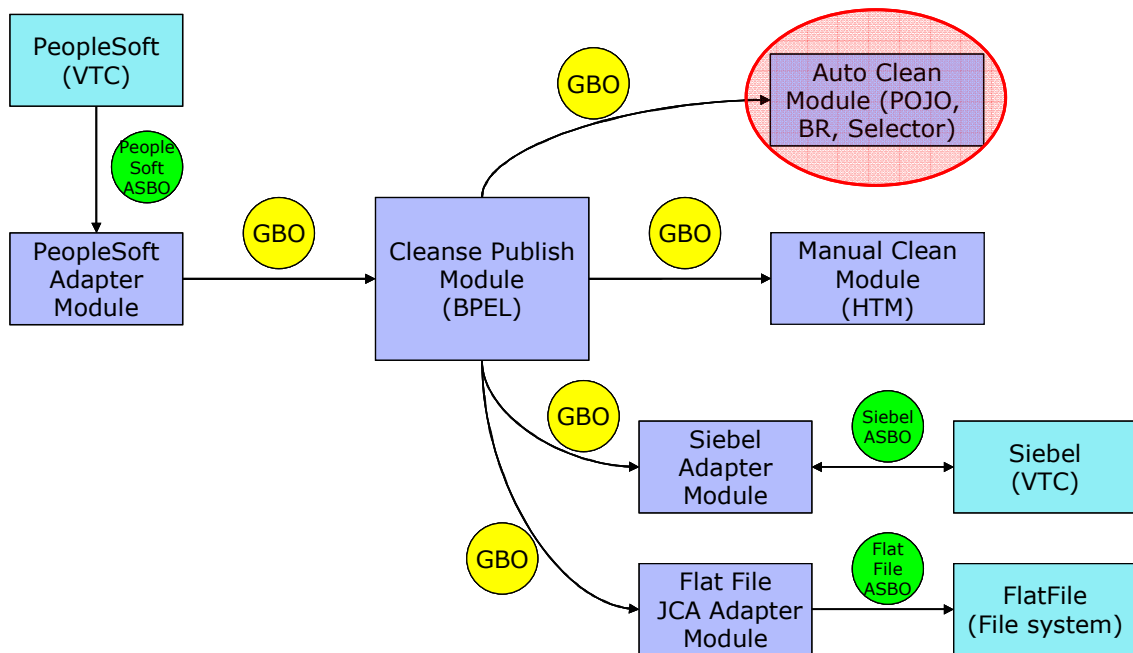
- WebSphere Integration Developer V6.0.2 installed
- WebSphere Process Server V6 test environment installed
- Sample code in the directory C:\Labfiles602 (Windows®) or /tmp/LabFiles602 (Linux®)

## What you should be able to do

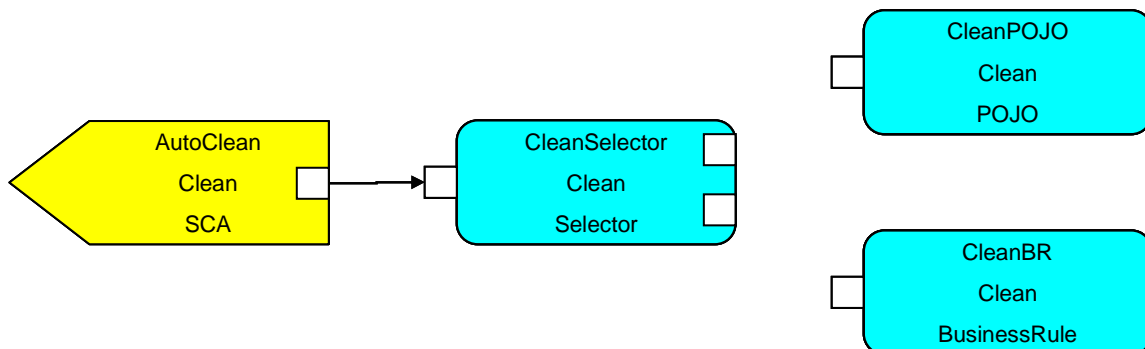
- Know how to import files into your integration projects
- Create and use Business Rule Groups, Business Rules, and Selectors
- Install and use the Business Rule Manager Web tool

## Introduction

In the diagram below, the CleansePublish Module in the middle acts as a Client. It sends a GBO or Generic Business Object to have the data in the GBO to be cleaned up for whoever is viewing that data by the AutoClean Module. The AutoClean Module is connected to the CleansePublish Module by way of an Export/Import. So in the lower diagram, the GBO is passed through a Selector called CleanSelector to determine which Target Implementation to use, either the CleanPOJO or the CleanBR business rule. This choice is determined by a date, either the current date or a date variable mined using Java™ or Xpath parameters.



## Description of the module



---

## 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	AIX®/Unix location
<WID_HOME>	C:\Program Files\IBM\WebSphere\ID\6.0	
<WPS_HOME>	<WID_HOME>\runtimes\bi_v6	
<LAB_FILES>	C:\Labfiles602	/tmp/Labfiles602
<TEMP>	C:\temp	/tmp
<SOLUTION>	C:\Labfiles602\AutoClean\Solution	/tmp/AutoClean/Solution
<WORKSPACE>	C:\Labfiles602\eXchange\AutoClean\workspace	/tmp/Labfiles602/eXchange/AutoClean/workspace

**Windows® users:** 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:

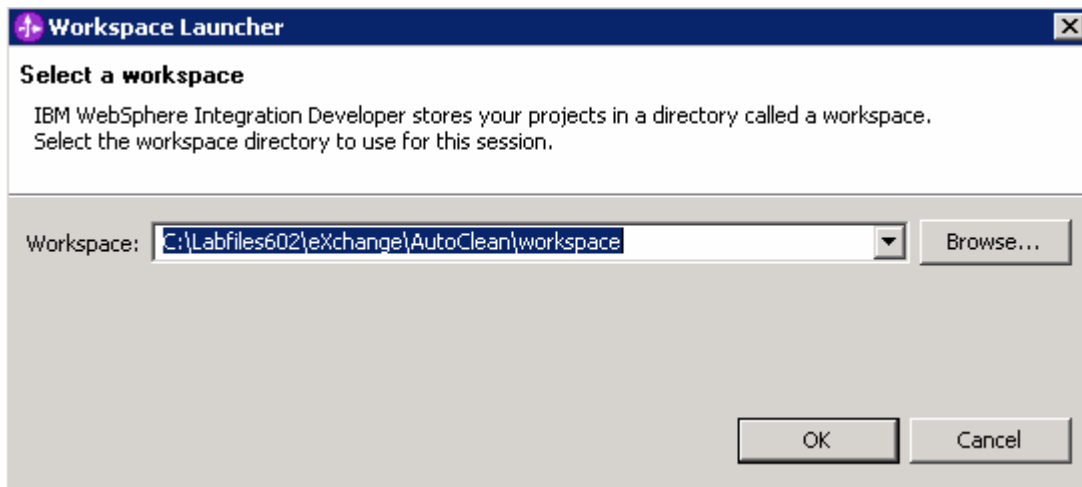
Reference Variable	Example: Remote Windows test server location	Example: Remote z/OS test server location	Input your values for the remote location of the test server
<SERVER_NAME>	server1	cl1sr01	
<WAS_HOME>	C:\Program Files\IBM\WebSphere\App Server	/etc/cl1cell/AppServerNode1	
<HOSTNAME>	localhost	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<sup>®</sup>, 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.

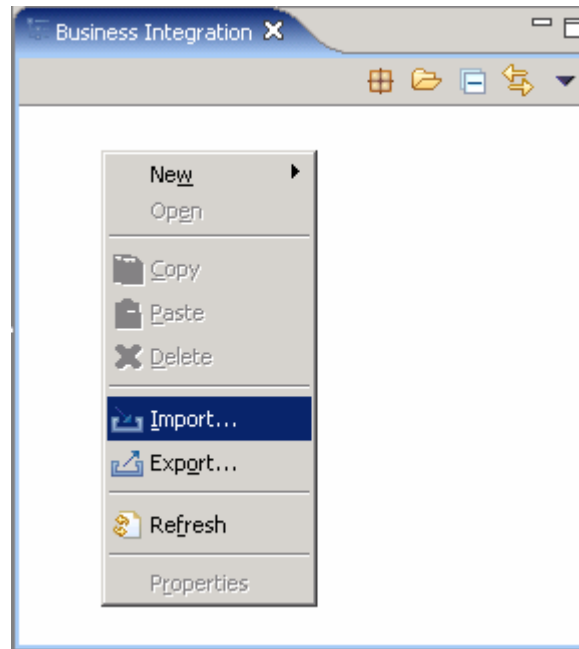
- \_\_\_ 1. Start WebSphere Integration Developer V6.0.2 with a workspace location of **<WORKSPACE>**, that is **<LAB\_FILES>\eXchange\AutoClean\workspace**
  - \_\_\_ a. From Windows<sup>®</sup> Explorer, navigate to the **<WID\_HOME>** directory and double click on wid.exe
  - \_\_\_ b. When prompted for workspace name, enter the value provided by the **<WORKSPACE>** variable for this lab and click **OK**



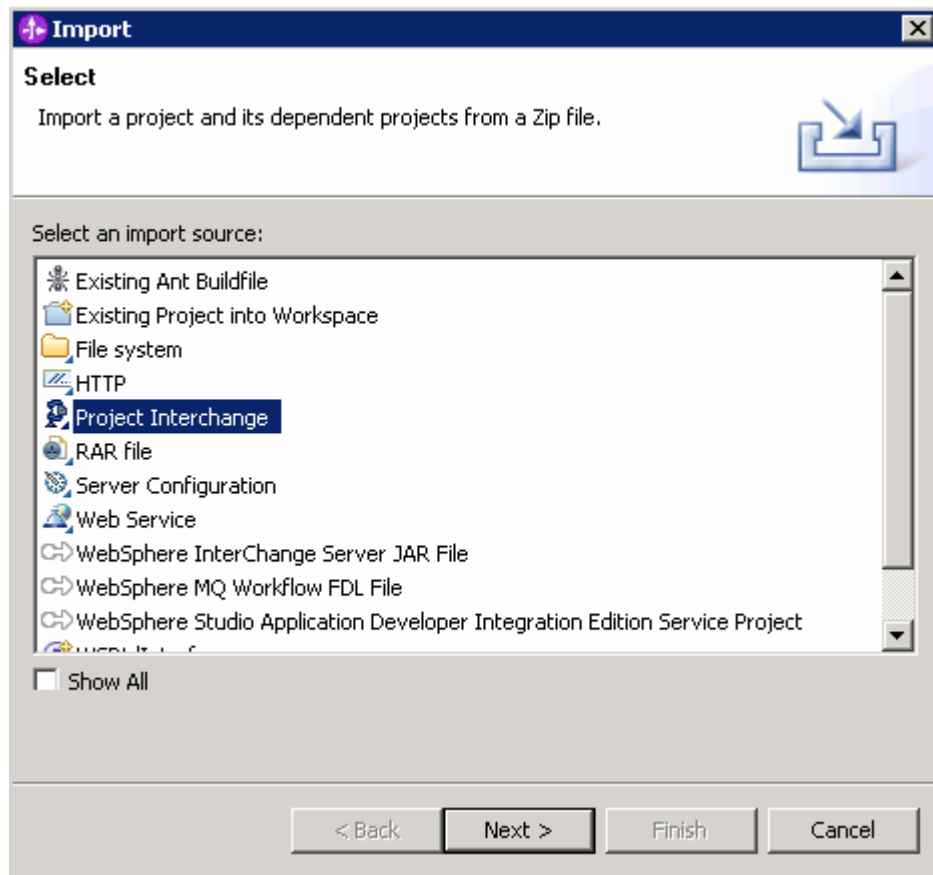
- \_\_\_ c. When WebSphere Integration Developer V6.0.2 opens, click the curved arrow at top right to **go to Business Integration perspective**.



- \_\_\_ 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

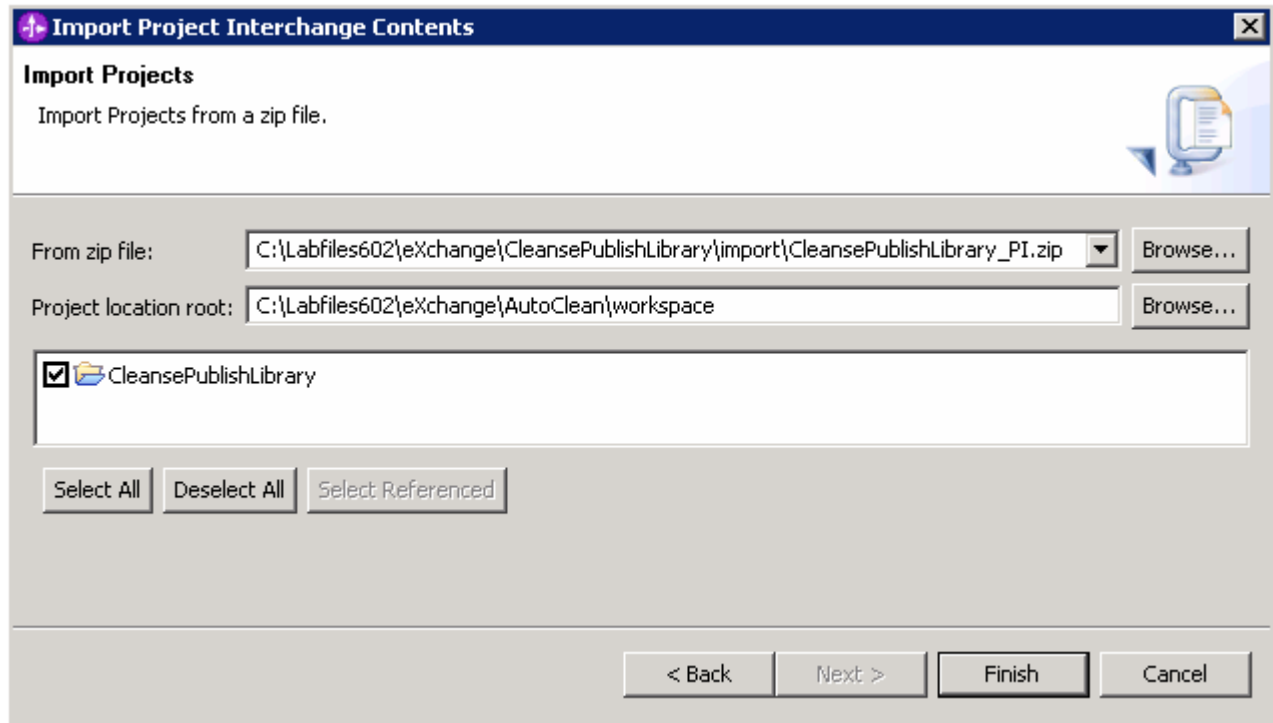


\_\_\_ b. Select **Project Interchange** listed in the import dialog.

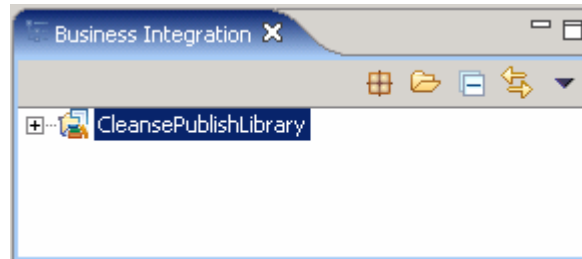


\_\_\_ c. Click **Next**

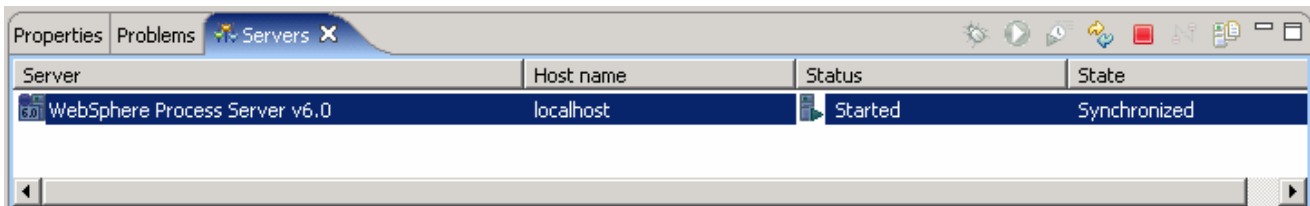
- \_\_\_ d. Click the **Browse** button for “**From zip file**” and navigate to <LAB\_FILES>eXchange\CleansePublishLibrary\import\CleansePublishLibrary\_PI.zip and hit **Open**



- \_\_\_ e. Select the check box next to **CleansePublishLibrary** and click **Finish**
- \_\_\_ f. Verify you have **CleansePublishLibrary** module listed in the Business Integration view



- \_\_\_ g. Verify you have WebSphere Process Server V6.0 listed in your Servers view





## Part 2: Create a module and import Java implementation

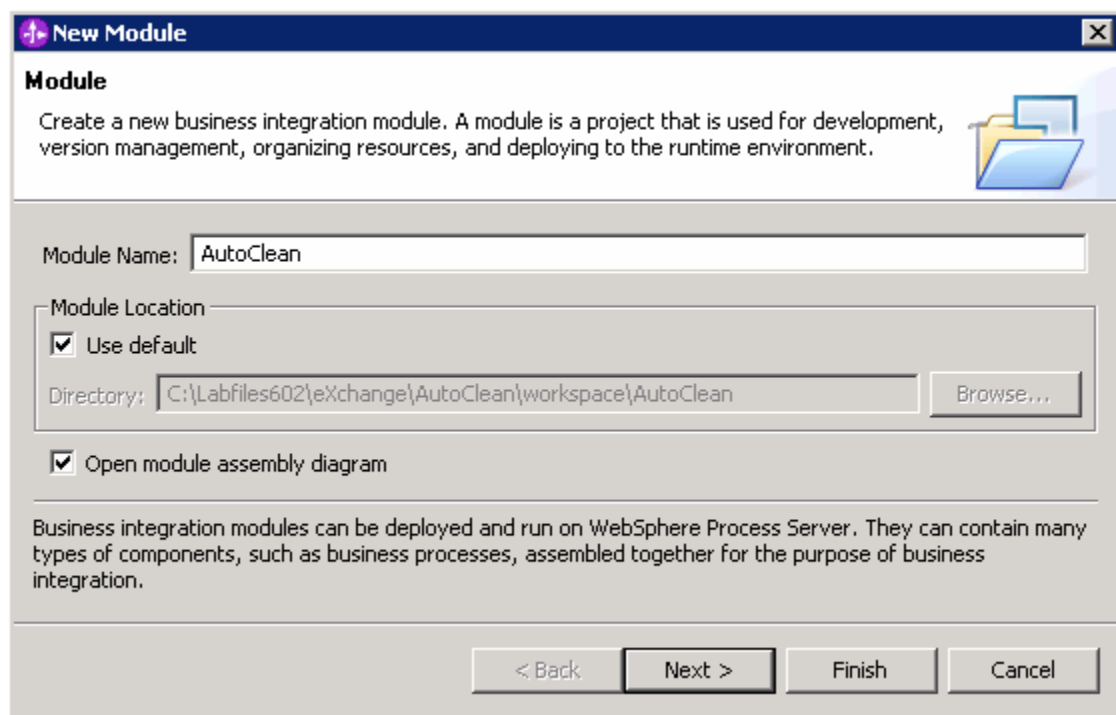
In this section of the lab, you will be creating a new module named AutoClean and eventually import a simple Java implementation. During the first iteration of the cleanse process a simple Java implementation will be created to handle the cleanse process. This means you need a Java implementation to start the process you are going to test. This process involves a Java implementation sending a message through a selector, where the selector chooses to use a POJO (plain old Java object) or a business rule given a certain date.

\_\_\_ 1. Create a new module named, **AutoClean**

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

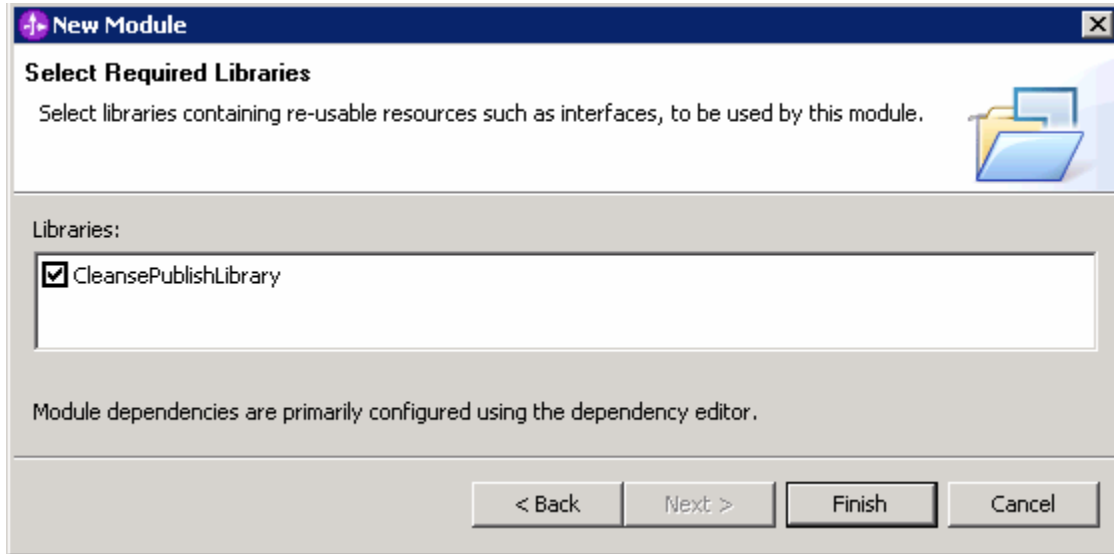


\_\_\_ b. Enter the Module Name as **AutoClean**



\_\_\_ c. Click **Next**

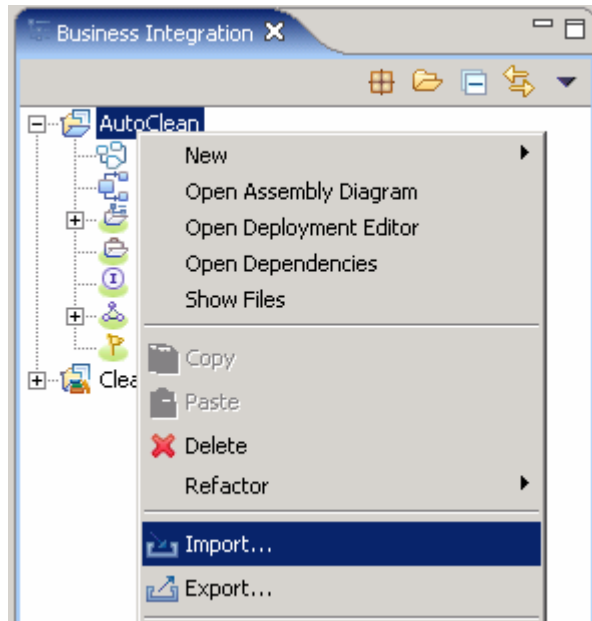
- \_\_\_ d. In the following panel, select the check box next to **CleansePublishLibrary**. This adds the CleansePublishLibrary as a dependant library for the AutoClean module



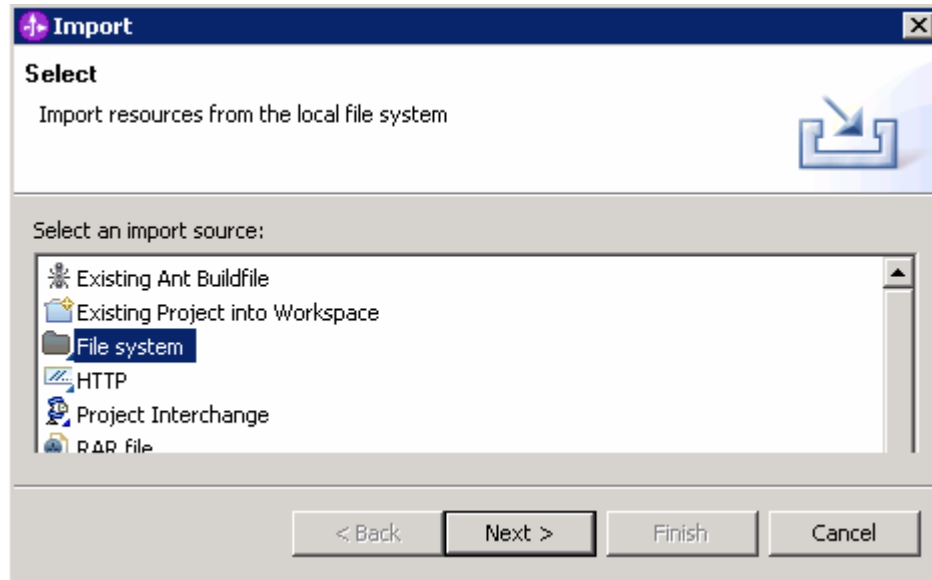
- \_\_\_ e. Click **Finish**

\_\_\_ 2. Import the POJO (Java implementation)

- \_\_\_ a. Right-click the **AutoClean** module in the Business Integration view and select **Import...** from the context menu

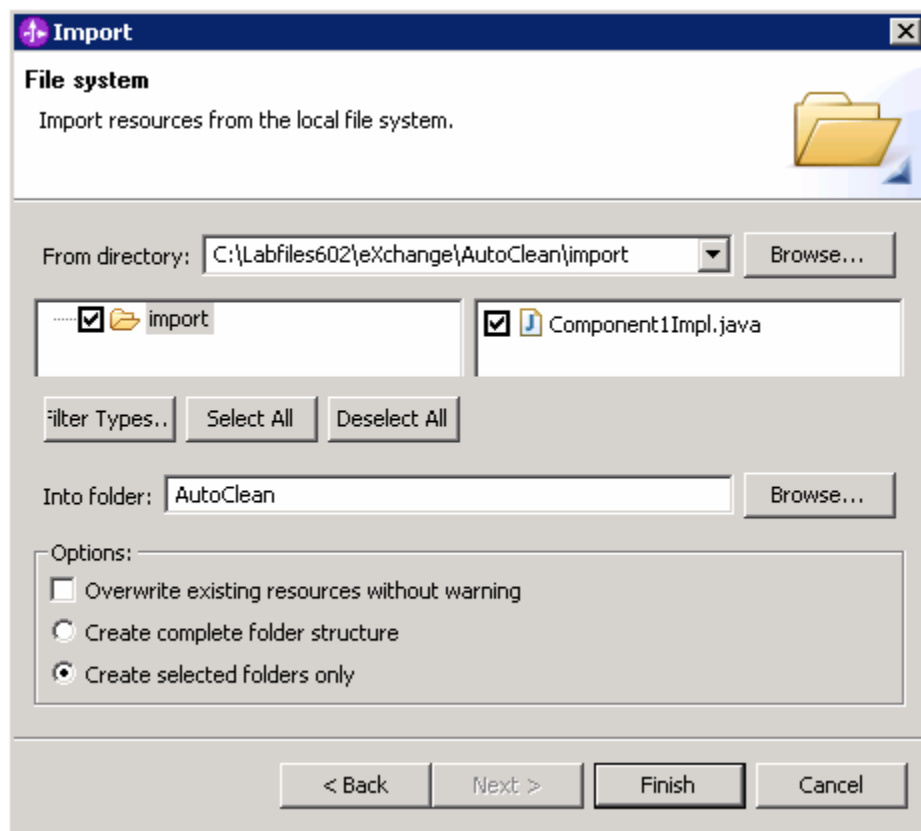


- \_\_\_ b. Select **File System** listed in the import dialog



\_\_ c. Click **Next**

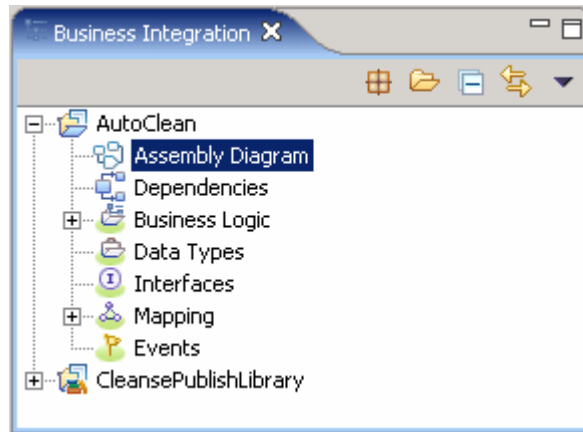
\_\_ d. Click the Browse button for the “**From Directory:**” and locate the Java implementation file, **Component1Impl.java** at **<LAB\_FILES>\eXchange\AutoClean\import**



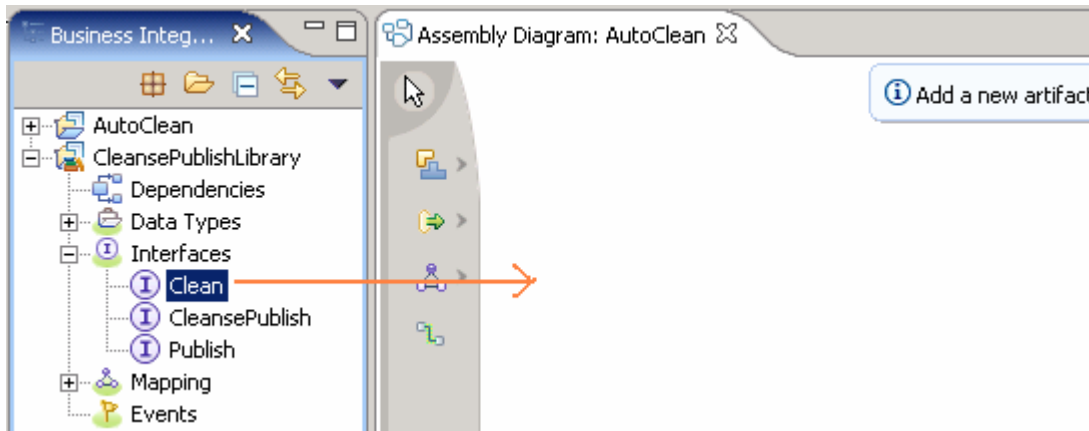
\_\_ e. Select the check box next to **Component1Impl.java** and ensure that the “**Into folder:**” field is populated with the folder name as **AutoClean** and click **Finish**

\_\_\_ 3. Create a Java Component

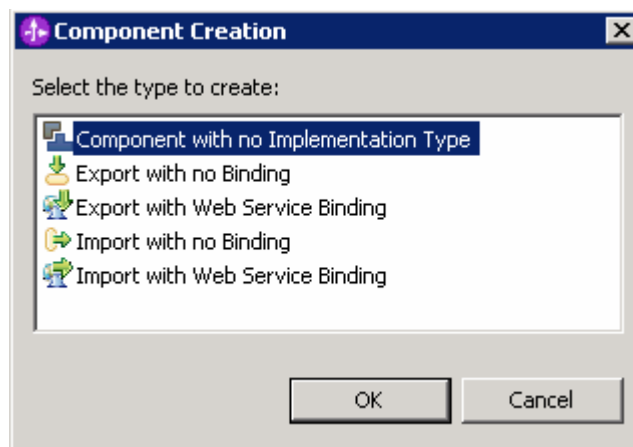
- \_\_\_ a. Expand **AutoClean** in the Business Integration view and double click on **Assembly Diagram** (Assembly Diagram) to open it in an Assembly Diagram editor



- \_\_\_ b. In the Business Integration view, expand **CleansePublishLibrary** → **Interfaces** and drag the **Clean** interface on to the Assembly Diagram. The Component Creation dialog pops-up



- \_\_\_ c. In the Component Creation dialog, select **Component with No Implementation Type**

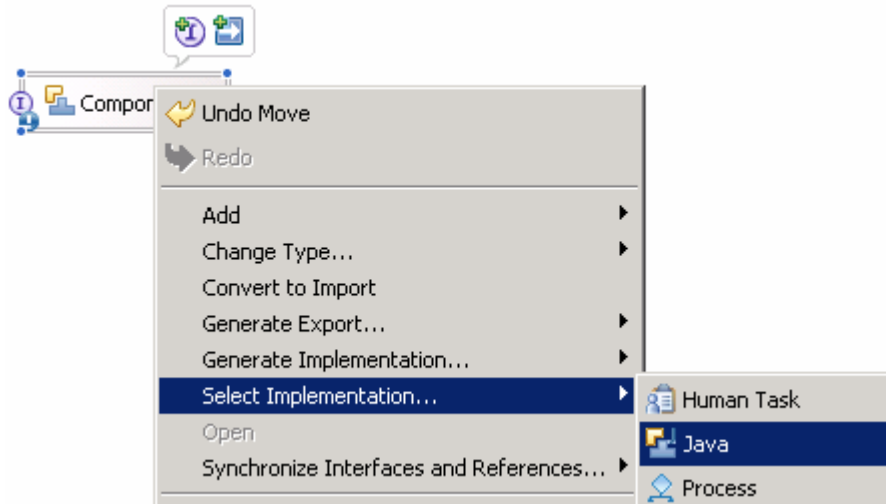


- \_\_\_ d. Click **OK**. A new component named **Component1** is created as shown below:



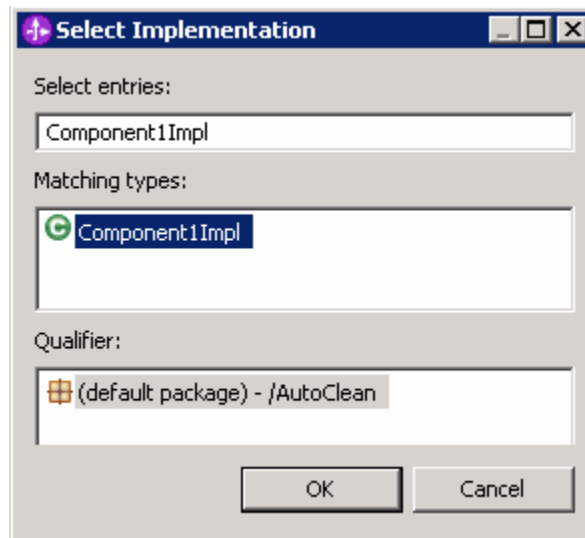
**Note:** Optionally, the Component1 display name can be changed to a more descriptive name. Right-click, select show in properties, the display name could be edited. For this lab, it is recommended you leave it as Component1

- \_\_\_ 4. Select the Java implementation for the new component created
  - \_\_\_ a. Right-click on **Component1** in the Assembly Diagram and choose **Select Implementation → Java** from the context menu



Note: Optionally, a new Java implementation could be generated using the **Generate Implementation...**

- \_\_\_ b. Type **Component1Impl** in the “**Select entries:**” field of the Pick Implementation dialog field, and select **Component1Impl** from the Matching types field



\_\_ c. Click **OK**

---

**Note:** What this does is, import the implementation class that uses the 'Clean' interface. Normally you would Generate Implementation and edit that code. However, in this lab there is no need to generate implementation since you use an existing implementation class in this lab.

---

\_\_\_\_ 5. Save all work by **File → Save All** or **Crtl + Shift + S**

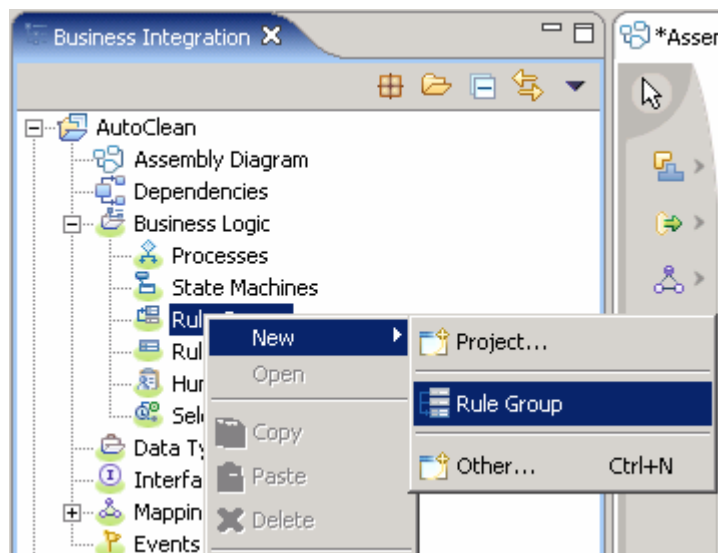
## Part 3: Create a business rule group

In this section, you will be creating a Business Rule Group. A Rule Group must be set up before creating a business rule implementation (Rule set or Decision Table). The job of the Rule Group is to group like business rules, and is the SCA component that is exposed as a service (seen by other SCA services). In this example, you will create a Ruleset to see business rule features.

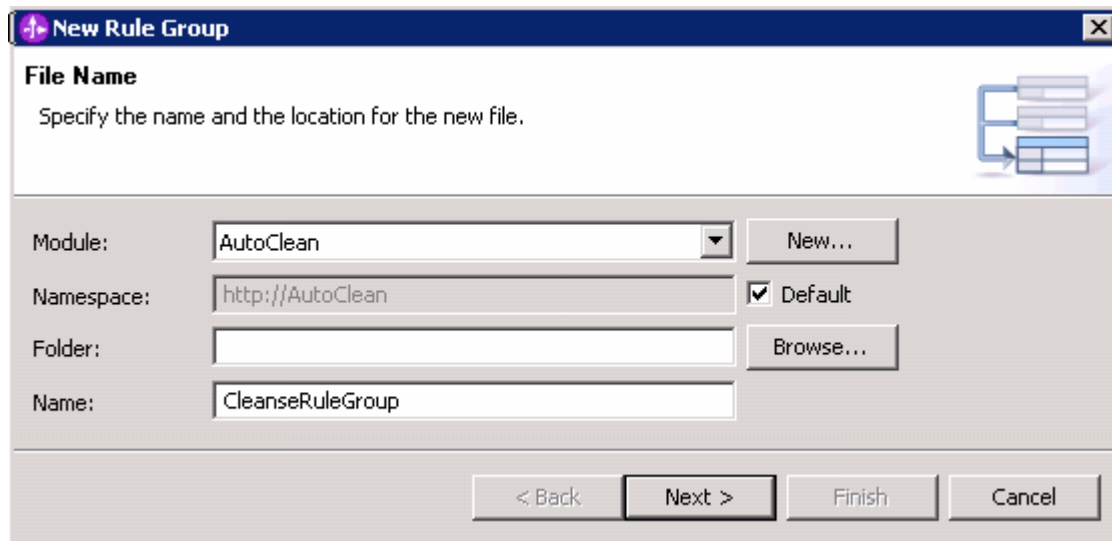
**Note:** You may receive errors in the Problems view as you create components. These errors will come and go as you save/build/finish building pieces.

### 1. Create a Rule Group

- a. In the Business Integration view, expand **AutoClean** → **Business Logic** → **Rule Groups**. Now right click over **Rule Groups** and select **New** → **Rule Group** from the context menu

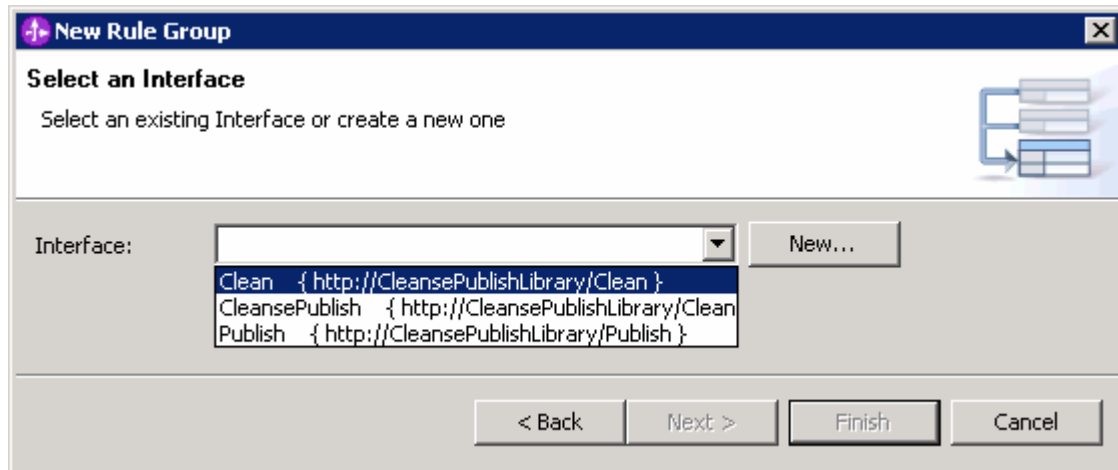


- b. In the New Rule Group dialog, enter **CleanseRuleGroup** for the Name

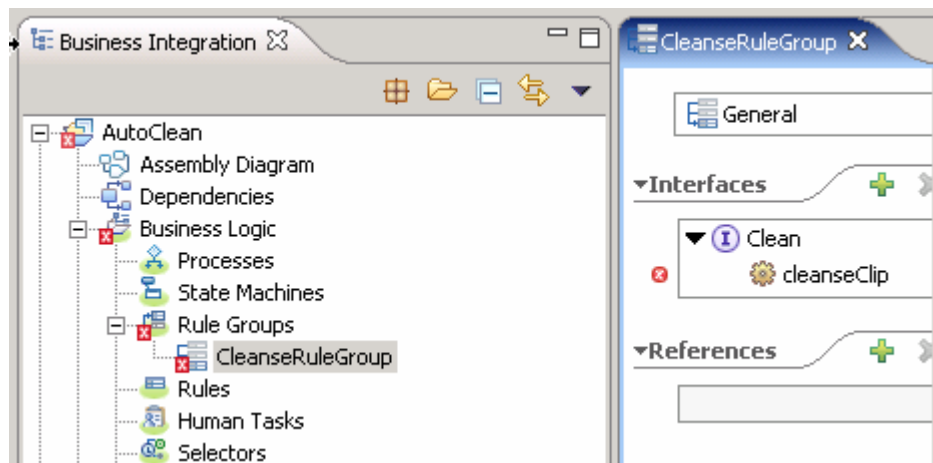


\_\_ c. Click **Next**

\_\_ d. In the following panel, select the interface as **Clean** from the drop down list



\_\_ e. Click **Finish**. The new rule group you had created is opened in the Rule Groups editor as shown below:



---

**Note:** At this time an error is displayed for CleanseClip operation and is resolved by specifying a rule destination.

---



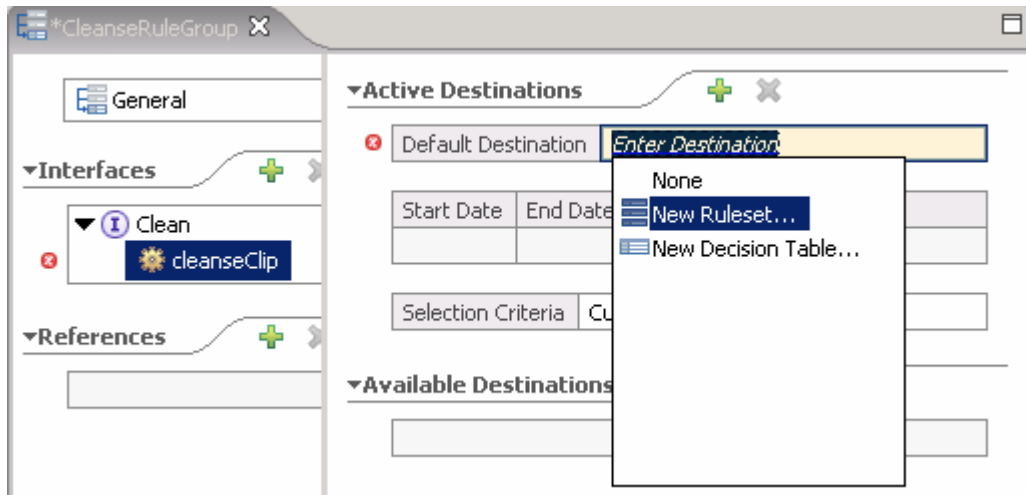
## Part 4: Create a business rule Set

In this section, you will be creating a Business Rule Set from a Rule Group. The Ruleset is one of the two implementations of a Business Rule, and the easiest one to show how a Business Rule works.

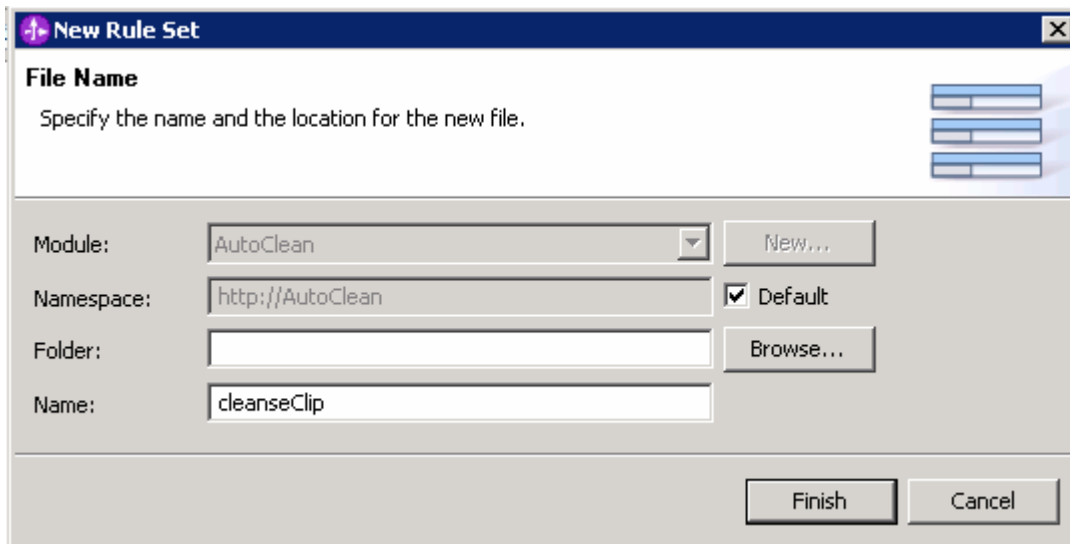
**Note:** You may receive errors/warnings in the Problems view as you create components.

\_\_\_ 1. Create a Rule Set

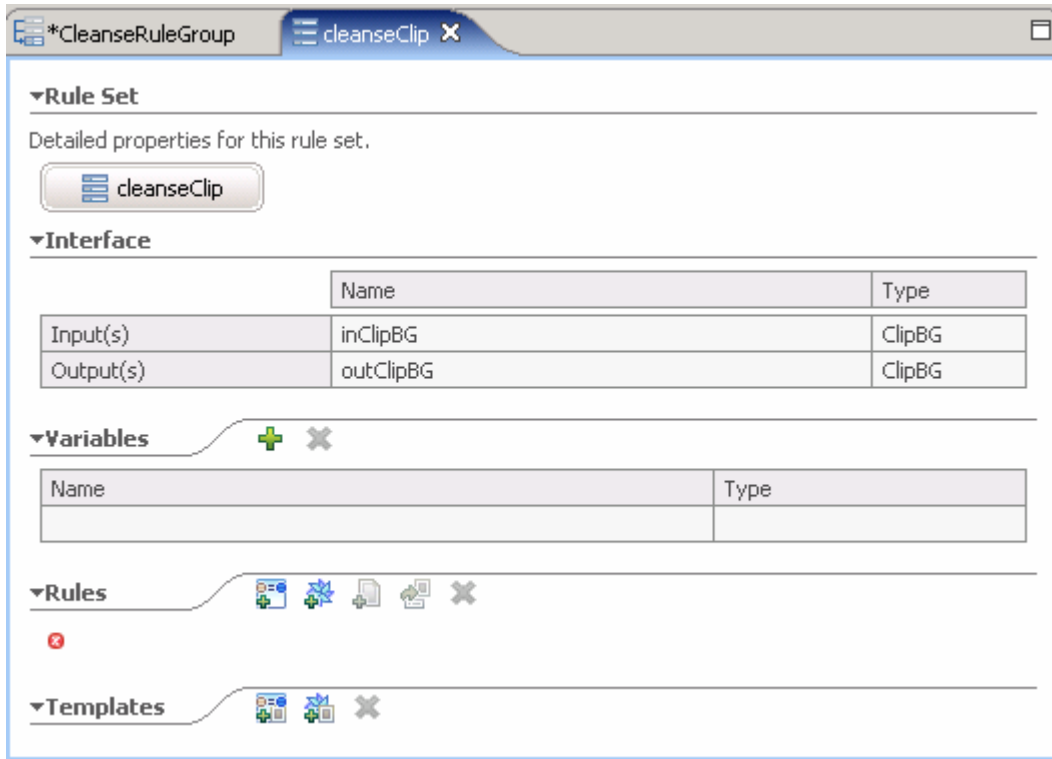
- \_\_\_ a. Select the **cleanseClip** under the **Clean** Interface, left-click on the **Enter Destination** for Default Destination and select **New Ruleset**. The New Rule Set dialog pops-up




- \_\_\_ b. Accept the defaults for the New Ruleset dialog



- \_\_\_ c. Click **Finish**. The Rule set editor for cleanseClip is opened as shown below:

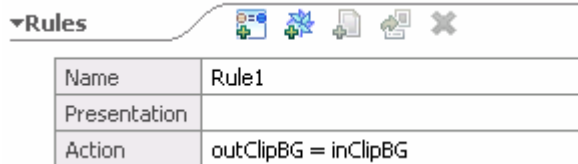


\_\_\_ d. Click the  icon under Variables section to add a local variable



\_\_\_ e. Enter **sizeRuleExecuted** for the name and select **boolean** for the Type

\_\_\_ f. Click the  icon to add the first Action Rule under the Rules section



\_\_\_ g. Click in the *Action* area and type **outClipBG = inClipBG**. You will notice that a Content Assist menu will appear when typing. Either use Content Assist or type in manually.

\_\_\_ h. While you are in the Action area, hit **Enter** at the end of the line to enter the initialization of the sizeRuleExecuted variable. Enter **sizeRuleExecuted = false** manually or use the use Content Assist utility. The Action Rule must look as shown below:

Rules	
Name	Rule1
Presentation	
Action	outClipBG = inClipBG sizeRuleExecuted = false

**Note:** There is no condition on Rule1 since it is an action, it will always be run first. The reason outClipBG gets inclipBG is to initialize every element in the Business Graph. The sizeRuleExecuted variable is initialized to "false" so that you can test for "true".

\_\_\_ i. Now click  icon under the Rules section to add an "If-Then" rule

Name	Rule2
Presentation	
If	Condition
Then	Action

\_\_\_ j. Click on the **Condition** and use Content Assist utility or type **inClipBG.Clip.size == "small"** manually for the **If** field. Click on **Action** in the Then section and use Content Assist utility or type **outClipBG.Clip.size = "small"** manually. Hit enter to advance to the next line and use Content Assist utility or type **sizeRuleExecuted = true**

Name	Rule2
Presentation	
If	inClipBG.Clip.size == "small"
Then	outClipBG.Clip.size = "small" sizeRuleExecuted = true




**Note:** In order to use the Business Rule Editor Web tool to modify the rules of an installed application, the rules must be expressed in the form of a template. This is why you are generating a template so later in this lab you can see it from the Business Rule Editor Web tool.

\_\_\_ k. Right-click on Rule 2, and select **Convert Rule to Template** from the context menu.

Name	Rule2	
Presentation		
If	inClipBG.	
Then	outClipBG. sizeRuleE	

Undo Expression  
Redo  
Delete Rule  
**Convert Rule to Template Ctrl+Alt+Shift+V**




\_\_\_ l. A template will be generated for Rule 2 as shown below:

▼ Templates   

Name	Template_Rule2		
Presentation	If inClipBG.Clip.size == '{param0}' then outClipBG.Clip.size = '{param1}' sizeRuleExecuted = '{param2}'		
Parameters	Name	Type	Constraint
	param0	string	None
	param1	string	None
	param2	boolean	None
If	inClipBG.Clip.size == param0		
Then	outClipBG.Clip.size = param1 sizeRuleExecuted = param2		

\_\_\_ m. Now under the **Templates** section, change the Template\_Rule2's action **sizeRuleExecuted = param2** to **sizeRuleExecuted = true** for the "Then" field




\_\_\_ n. Now delete the **param2** of type boolean generated in the Parameters filed from the template since it is always set to true when the rule is ran. The deletion can be done by right clicking over the parameter and selecting **Delete Parameter** from the context menu

▼ Templates   

Name	Template_Rule2		
Presentation	If inClipBG.Clip.size == '{param0}' then outClipBG.Clip.size = '{param1}' sizeRuleExecuted = '{param2}'		
Parameters	Name	Type	Constraint
	param0	string	None
	param1	string	None
	param2	boolean	None
If	inClipBG.Clip.size == param0		
Then	outClipBG.Clip.size = param1 sizeRuleExecuted = true		

Undo Expression  
Redo Delete Parameter  
Delete Rule  
**Delete Parameter**

\_\_\_ o. As a result of the deletion of param2, the place holder for the Boolean variable {2} is deleted from the presentation filed. Now delete the **sizeRuleExecuted=** from the Presentation filed

▼ Templates   

Name	Template_Rule2		
Presentation	If inClipBG.Clip.size == '{param0}' then outClipBG.Clip.size = '{param1}' sizeRuleExecuted =		
Parameters	Name	Type	Constraint
	param0	string	None
	param1	string	None
If	inClipBG.Clip.size == param0		
Then	outClipBG.Clip.size = param1 sizeRuleExecuted = true		

\_\_\_ p. The Presentation field can be modified to present the rule in a natural language form. The Presentation field will be displayed in the Business Rules Editor Web tool. Change the Presentation field in Template\_Rule2 to: **If the incoming clip size is: {param0}, set the clip size to the standard value: {param1}**


Presentation	If the incoming clip size is : '{param0}', set the clip size to the standard value : '{param1}'
--------------	---

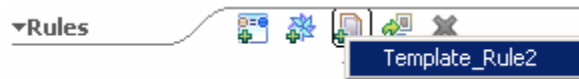
**Note:** Modifying the text in the Presentation field is important because this is all the Business Analyst will see when editing the Business Rules in the Business Rule Editor Web tool.

\_\_ q. The final Template\_Rule2 must look like as shown below:

▼ Templates			
Name	Template_Rule2		
Presentation	If the incoming clip size is : '{param0}', set the clip size to the standard value : '{param1}'		
Parameters	Name	Type	Constraint
	param0	string	None
	param1	string	None
If	inClipBG.Clip.size == param0		
Then	outClipBG.Clip.size = param1 sizeRuleExecuted = true		

\_\_ r. Creating a few rules from the template

- 1) Click  icon under the Rules section and select the only option... **Template\_Rule2**



- 2) In the Presentation field enter **regular** for the incoming clip size value leave the output value of the clip size as blank as shown below:

Presentation	If the incoming clip size is : <input type="text" value="Enter Value"/> , set the clip size to the standard value : <input type="text" value="Enter Value"/>
--------------	--

To


Presentation	If the incoming clip size is : <input type="text" value="regular"/> , set the clip size to the standard value : <input type="text"/>
--------------	--

- 3) Repeat the above 2 steps ( 1 and 2) to create **4 new rules** from Template\_Rule2.

**Note:** <a blank>, denotes a null value. Delete 'Enter Value' in the output area.

- a) With values **regular** and <a blank>
- b) With values **medium** and <a blank>
- c) With values **large** and **large**
- d) With values **jumbo** and **large**

\_\_ s. Create a final rule to catch any other types of input

- 1) Click  icon under the Rules section to add an "If-Then" rule
- 2) Click in the "If" field and select or type **sizeRuleExecuted == false**

- 3) Click in the 'Then' filed and select or type **outClipBG.Clip.size = "?" + inClipBG.Clip.size**.  
This outputs a question mark before the input text to show that a rule was not fired.

\_\_ t. Here is how the final view of the Ruleset added must look like:

▼Rules	
Name	Rule1
Presentation	
Action	outClipBG = inClipBG sizeRuleExecuted = false
Name	Rule2
Template	Template_Rule2
Presentation	If the incoming clip size is : <input type="text" value="small"/> , set the clip size to the standard value : <input type="text" value="small"/>
Name	Rule3
Template	Template_Rule2
Presentation	If the incoming clip size is : <input type="text" value="regular"/> , set the clip size to the standard value : <input type="text"/>
Name	Rule4
Template	Template_Rule2
Presentation	If the incoming clip size is : <input type="text" value="medium"/> , set the clip size to the standard value : <input type="text"/>
Name	Rule5
Template	Template_Rule2
Presentation	If the incoming clip size is : <input type="text" value="large"/> , set the clip size to the standard value : <input type="text" value="large"/>
Name	Rule6
Template	Template_Rule2
Presentation	If the incoming clip size is : <input type="text" value="jumbo"/> , set the clip size to the standard value : <input type="text" value="large"/>
Name	Rule7
Presentation	
If	sizeRuleExecuted == false
Then	outClipBG.Clip.size = "?" + inClipBG.Clip.size

▼Templates				
Name	Template_Rule2			
Presentation	If the incoming clip size is : <input type="text" value="param0"/> , set the clip size to the standard value : <input type="text" value="param1"/>			
Parameters	Name	Type	Constraint	+
	param0	string	None	
	param1	string	None	
If	inClipBG.Clip.size == param0			
Then	outClipBG.Clip.size = param1 sizeRuleExecuted = true			


\_\_ u. Save all work by **File → Save All** or **Ctrl + Shift + S**.


---

**Note:** There should not be any errors displayed at this time in the problems view.

---

\_\_\_\_ 2. Add the Rule Group, that is the CleanseRuleGroup to the Assembly Editor.

1) Expand **AutoClean** in the Business Integration view and double click on **Assembly Diagram** ( **Assembly Diagram**) to open it in an Assembly Diagram editor

2) Expand **AutoClean → Business Logic → Rule Groups** in the Business Integration view, drag  **CleanseRuleGroup** onto the assembly editor.

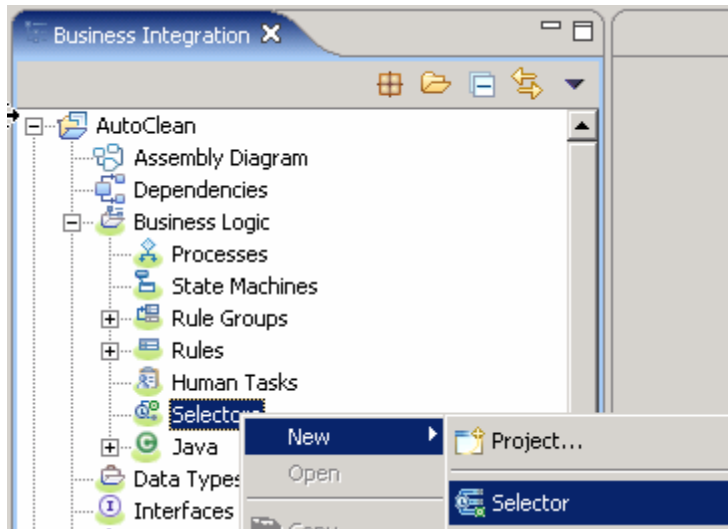


\_\_\_\_ 3. Save all work by **File → Save All** or **Crtl + Shift + S**

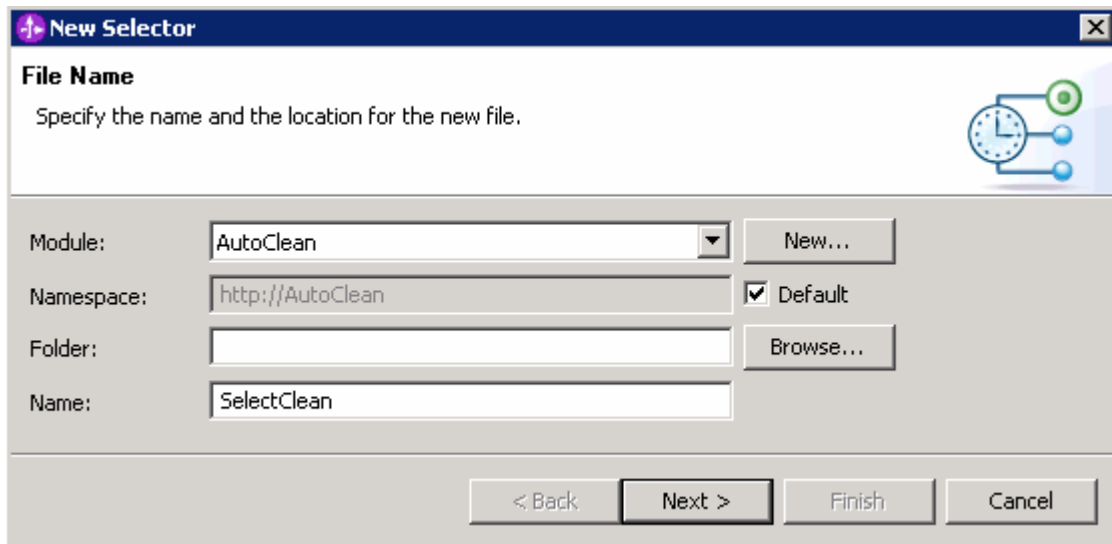
## Part 5: Create a selector

A simple selector will be created to invoke the simple Java business rule, which at some time will switch to use the business rule. You created and imported the component destinations first so that creating the selector is easier since you already have the destination components in the drop down list.

- \_\_\_ 1. Create a new selector
  - \_\_\_ a. Expand **AutoClean** → **Business Logic** → **Selectors** in the Business Integration view
  - \_\_\_ b. Right click on **Selectors** and select **New** → **Selector** from the context menu



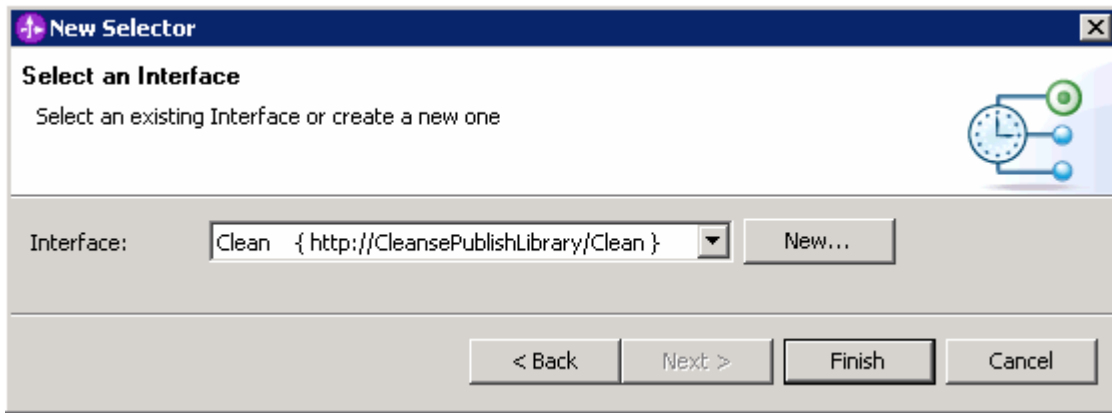
- \_\_\_ 2. In the new Selector dialog, enter **SelectClean** for the Name




- \_\_\_ 3. Click **Next**



- \_\_\_ 4. In the following panel, select **Clean** as the interface



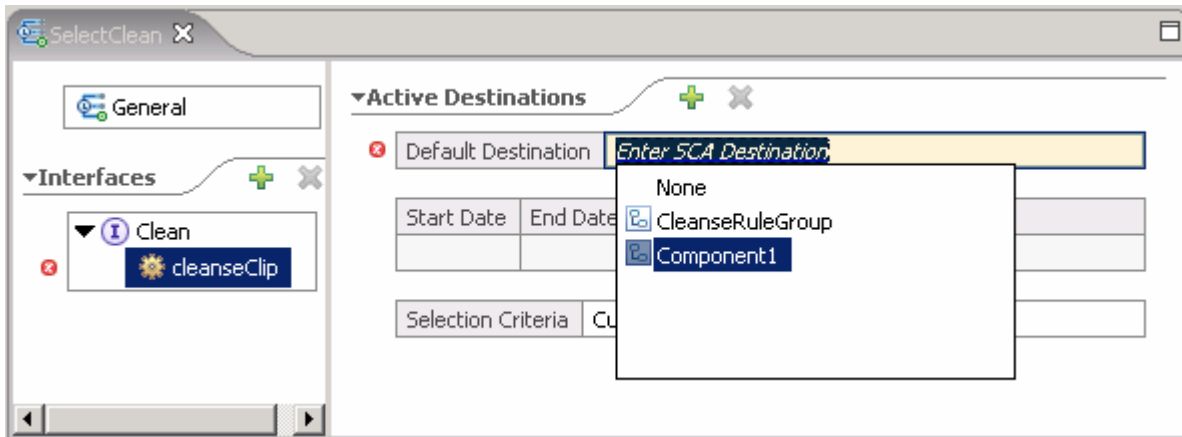
- \_\_\_ 5. Click **Finish**. The Selector you had created is opened in a Selector editor
- \_\_\_ 6. Now click on the operator, that is cleanseClip icon  **cleanseClip** under the Clean Interface to open Selector properties


---

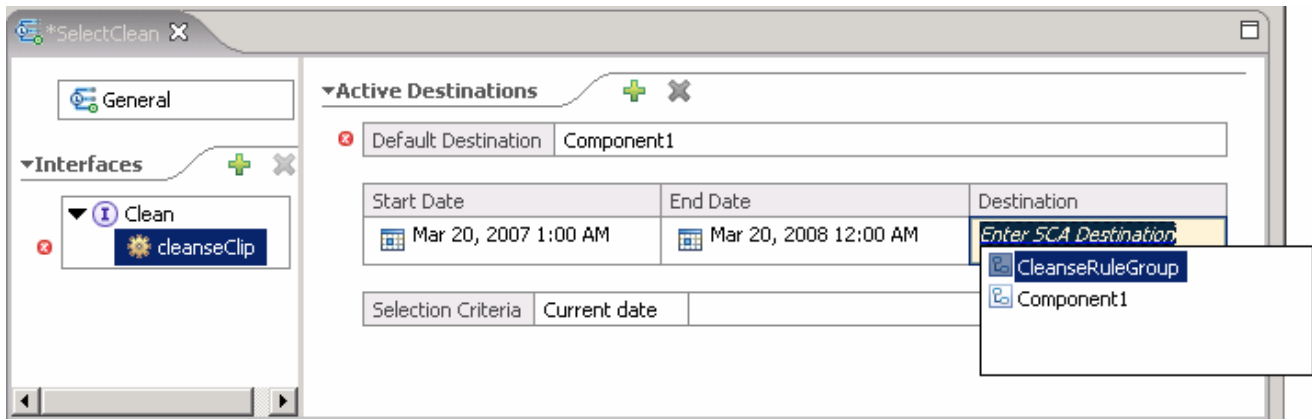
**Note:** There must be errors displayed at this time in the problems view.

---

- \_\_\_ 7. Click over **Enter SCA Destination** for the Default Definition filed and select **Component1**



- \_\_\_ 8. Click on the **plus icon**  to add date selection entry.
- \_\_\_ 9. Change the time to add **1 hour** to the current time under the Start Date field. You will be adding an hour so that in 1 hour, the selector will switch over from using Component1 (the POJO) to the Business Rule.
- \_\_\_ 10. Click on the **Enter SCA Destination** and select **CleanseRuleGroup**



\_\_\_ 11. Save all work by **File → Save All** or **Ctrl + Shift + S**

**Note:** There should not be any errors displayed at this time in the problems view.

\_\_\_ 12. Drag the Selector onto the Assembly Diagram

- \_\_\_ a. Expand **AutoClean** in the Business Integration view and double click on **Assembly Diagram** (Assembly Diagram) to open it in an Assembly Diagram editor
- \_\_\_ b. Expand **AutoClean → Business Logic → Selectors** in the Business Integration view
- \_\_\_ c. Now drag SelectClean onto the Assembly Editor
- \_\_\_ d. Your Assembly Editor should look like the one shown below:




**Note:** You cannot wire the Selector to the destination components. However, the Selector is already wired to the two target components - Component1 (the POJO) and the CleanseRuleGroup (the Business Rule). The selector is using dynamic wires to point at Component1 and the CleanseRuleGroup in the background. Meaning the Selector is already “wired” to those destination SCA components. Any editing of the Target Implementations would need the Selector Editor and not the Assembly Editor.


\_\_\_ e. Save all work by **File → Save All** or **Ctrl + Shift + S**

## Part 6: Create an export on assembly editor

In order for these components to be accessed from other modules, you must add an Export. The Export acts like a door for other modules to access the modules inside. Export = Allow others to use this module's services.

\_\_\_ 1. Expand **AutoClean** in the Business Integration view and double click on **Assembly Diagram** (  **Assembly Diagram** ) to open it in an Assembly Diagram editor

\_\_\_ 2. Add an Export to the **SelectClean** selector in the Assembly Diagram.


\_\_\_ a. Click on **Export** icon from Assembly Diagram tray (  )

\_\_\_ b. Click or drag icon to left side of the **SelectClean** selector

\_\_\_ c. Change the **Display Name** and **Name** from **Export1** to **AutoCleanExport**



\_\_\_ 3. Add the Clean interface to the Export. The reason you need to add an interface is to define the inputs and outputs of the module, so requests from clients know what data types the module uses

\_\_\_ a. Hover over export or click on Export to make the add interface icon appear. Click "I" icon (  ) to add an interface



\_\_\_ b. Select the **Clean** interface when prompted by the Add Interface wizard and click **OK**

\_\_\_ 4. Wire the Export to the Selector. Wiring the Export to the Selector creates the bridge that allows other SCA services to use the Selector

\_\_\_ a. Click on **AutoCleanExport** and drag a wire to **SelectClean**

\_\_\_ b. Your Assembly Editor should now look something like the picture below:



\_\_\_\_ 5. Save all work by **File → Save All** or **Crtl + Shift + S**

You now had created an export so this Selector/Business Rule module can be accessed and used by the other modules.

---

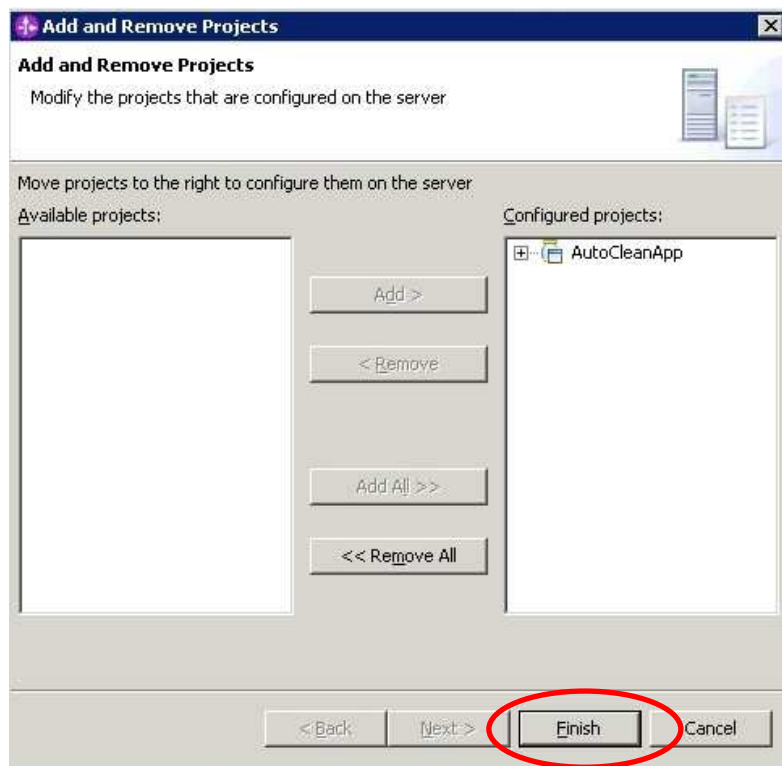
## Part 7: Publish the AutoCleanApp

In order to see the Business Rules with the Business Rules Editor Web tool and to test the Business Rules using the Component Test feature in WebSphere Integration Developer, you must publish the application to the WebSphere Process Server. Along with the tools there is a full version of the WebSphere Process Server that you can use to auto-publish to. You could manually install the application using the server Administrative Console. However, this can be accomplished much quicker by adding the project to the server from the WebSphere Integration Developer.

1. If using a remote testing environment, follow the directions provided in [Task: Adding Remote Server to Test Environment](#) at the end of this document to add a server to the 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, right-click on WebSphere Process Server V6.0 and select Start.

2. Right click on the WebSphere Process Server listed in the Servers View and select **Add and Remove Projects** from the context menu
3. Click the **Add** button to add the AutoCleanApp to the server.



4. Click **Finish**

---

**Note:** The WebSphere Integration Developer will start the server automatically to publish the application if the server is not already started.

---

## Update business rules using the business rule editor Web tool

The business rule editor Web tool allows a business analyst or a less technical/more business minded person to make business rule changes

The business rule manager Web tool is used to edit, delete and create business rules that have been "templated" and are running on the WebSphere Process Server. You can only see business rules that have been converted to a template. For instance, business rules converted in the WebSphere Integration Developer can be seen in the business rule manager Web tool. The business rule editor Web tool allows a business analyst or less technical, more business-minded person to make business rule changes without having to restart the server or the application. The Web tool is part of the runtime server, and not part of the WebSphere Integration Developer.

\_\_\_\_ 1. Ensure that the WebSphere Process Server test server is completely started at this time

---

**Note:** The following files, included in the runtime, are needed to install the business rule editor Web tool:

1) installBRManager.jacl (located at <WPS\_HOME>bin)

There are **three** requirements to use the jacl file.

- Any security setting of Process Server is to be disabled.
- Code page of command prompt or shell is set to be US environment.
- WebSphere Process Server must be started.

2) brmanager.war (located at <WPS\_HOME>/installableApps) .

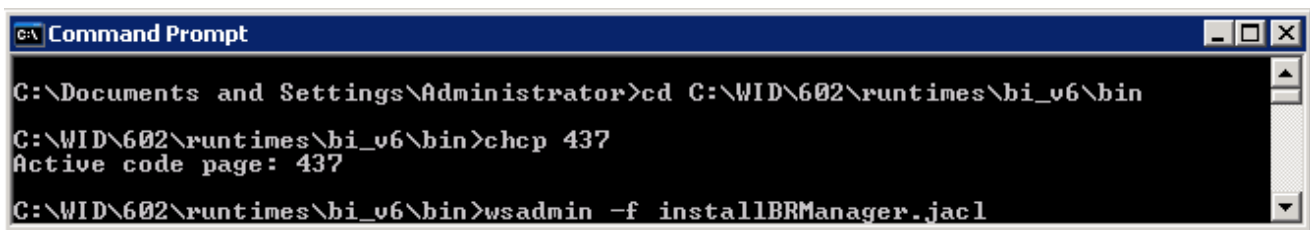
---

\_\_\_\_ 2. Run shell (for LINUX) or open a command prompt (for Windows®) and change directory to <WPS\_HOME>/bin (ex: cd C:\WID\runtimes\bi\_v6\bin )

\_\_\_\_ 3. Make sure the code page is set to US by typing **chcp 437** on Win32 machines, **LANG=C** then **LC\_CTYPE=C** for UNIX machines

\_\_\_\_ 4. Run the following command

Win32 : wsadmin.bat -f installBRManager.jacl  
 UNIX : ./wsadmin.sh -f <full\_path>/installBRManager.jacl



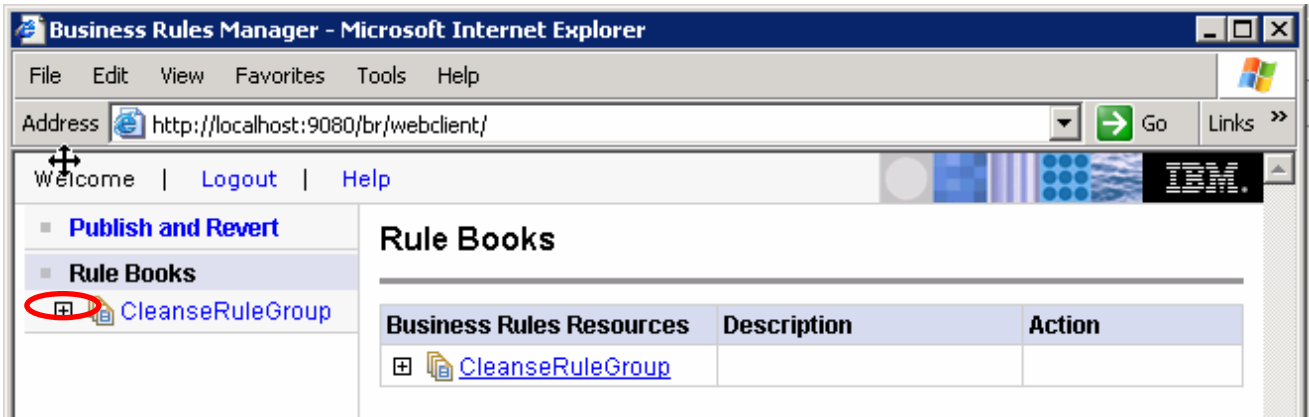
```

C:\Documents and Settings\Administrator>cd C:\WID\602\runtimes\bi_v6\bin
C:\WID\602\runtimes\bi_v6\bin>chcp 437
Active code page: 437
C:\WID\602\runtimes\bi_v6\bin>wsadmin -f installBRManager.jacl
  
```

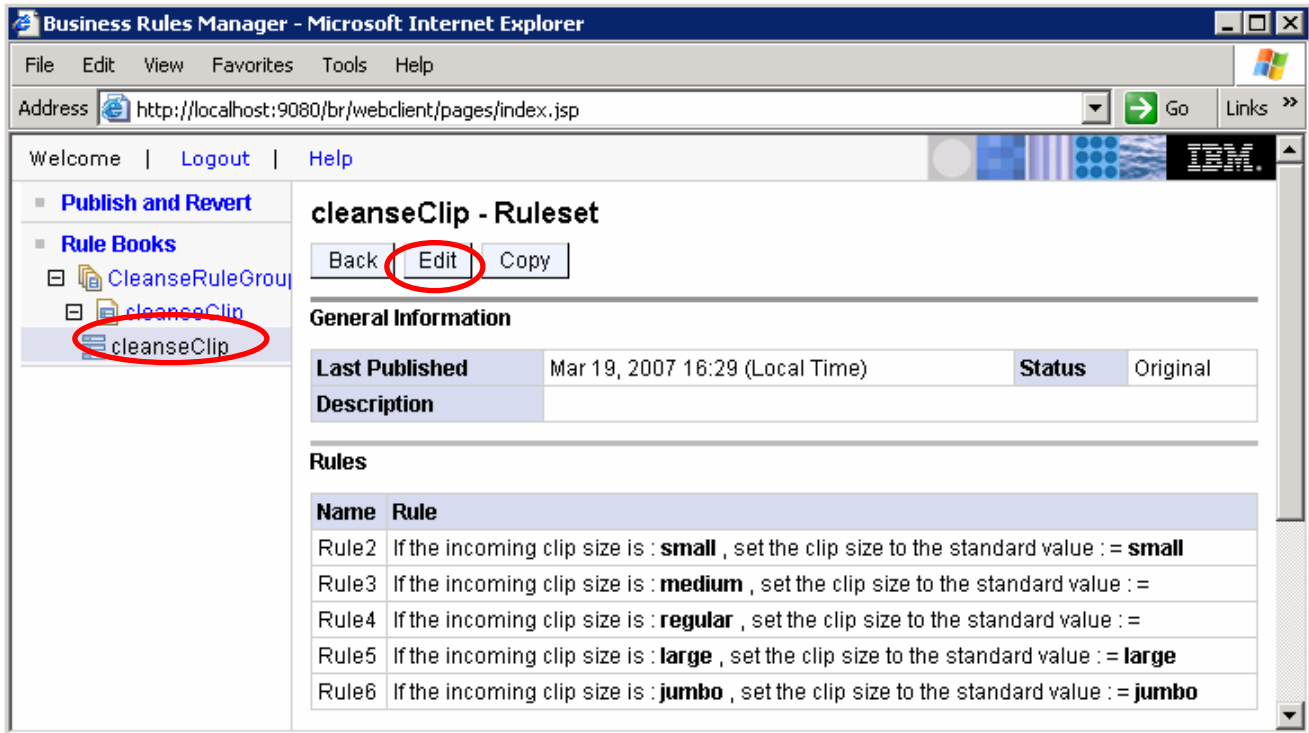
\_\_\_\_ 5. The business rule Web tool will now be installed and started. You will be able to see the application in the Administrative Console under Enterprise Applications (application is named as BRManager).

\_\_\_\_ 6. Access the Web tool by opening a browser and the following URL:

<http://<HOSTNAME>:<PORT>/br/webclient/pages/index.jsp>



7. In the Left navigation frame, expand the CleanseRuleGroup and click the cleanseClip as shown below:



8. The above are the rules that were published with the AutoClean Application. Click the **Edit** button at top to edit the templates you set up

**Note:** With the Web tool, you can change template values, move rules up or down for order of execution (top first, bottom last), and the ability to delete a templated rule. If the rule is not templated, you or a business analyst will not be able to see those rules in the Web tool.

The screenshot shows the Business Rules Manager interface in Microsoft Internet Explorer. The browser address bar shows `http://localhost:9080/br/webclient/pages/index.jsp`. The page title is "Business Rules Manager - Microsoft Internet Explorer". The main content area is titled "Edit Mode: cleanseClip - Ruleset". Below the title are "Save" and "Cancel" buttons and a "Messages:" field. The "General Information" section contains a table with the following data:

Last Published	Mar 19, 2007 16:29 (Local Time)	Status	*Original
Description			

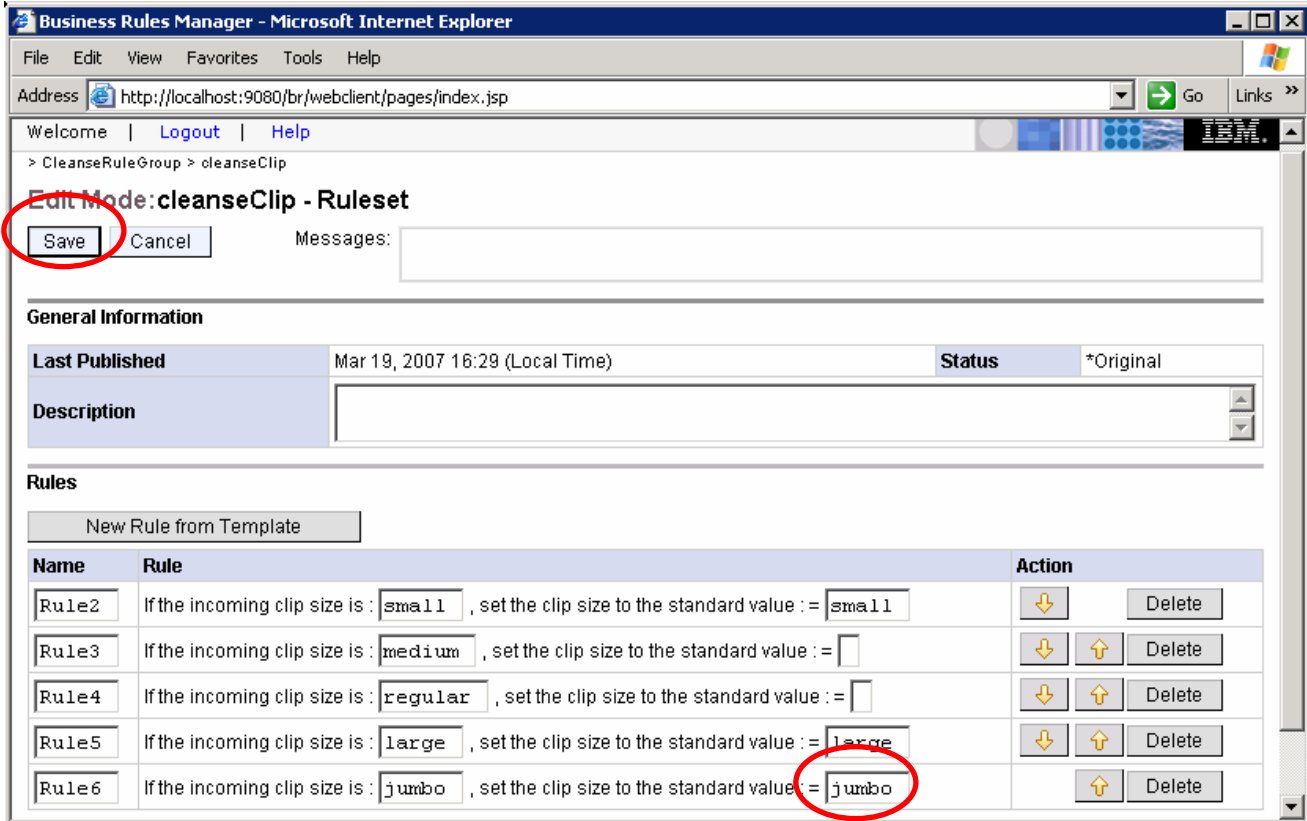
The "Rules" section features a "New Rule from Template" button and a table of rules:

Name	Rule	Action
Rule2	If the incoming clip size is : <input style="width: 50px;" type="text" value="small"/> , set the clip size to the standard value : = <input style="width: 50px;" type="text" value="small"/>	<input type="button" value="↓"/> <input type="button" value="Delete"/>
Rule3	If the incoming clip size is : <input style="width: 50px;" type="text" value="medium"/> , set the clip size to the standard value : = <input style="width: 50px;" type="text" value=""/>	<input type="button" value="↓"/> <input type="button" value="↑"/> <input type="button" value="Delete"/>
Rule4	If the incoming clip size is : <input style="width: 50px;" type="text" value="regular"/> , set the clip size to the standard value : = <input style="width: 50px;" type="text" value=""/>	<input type="button" value="↓"/> <input type="button" value="↑"/> <input type="button" value="Delete"/>
Rule5	If the incoming clip size is : <input style="width: 50px;" type="text" value="large"/> , set the clip size to the standard value : = <input style="width: 50px;" type="text" value="large"/>	<input type="button" value="↓"/> <input type="button" value="↑"/> <input type="button" value="Delete"/>
Rule6	If the incoming clip size is : <input style="width: 50px;" type="text" value="jumbo"/> , set the clip size to the standard value : = <input style="width: 50px;" type="text" value="large"/>	<input type="button" value="↑"/> <input type="button" value="Delete"/>

The "large" value in the action field of Rule6 is circled in red.

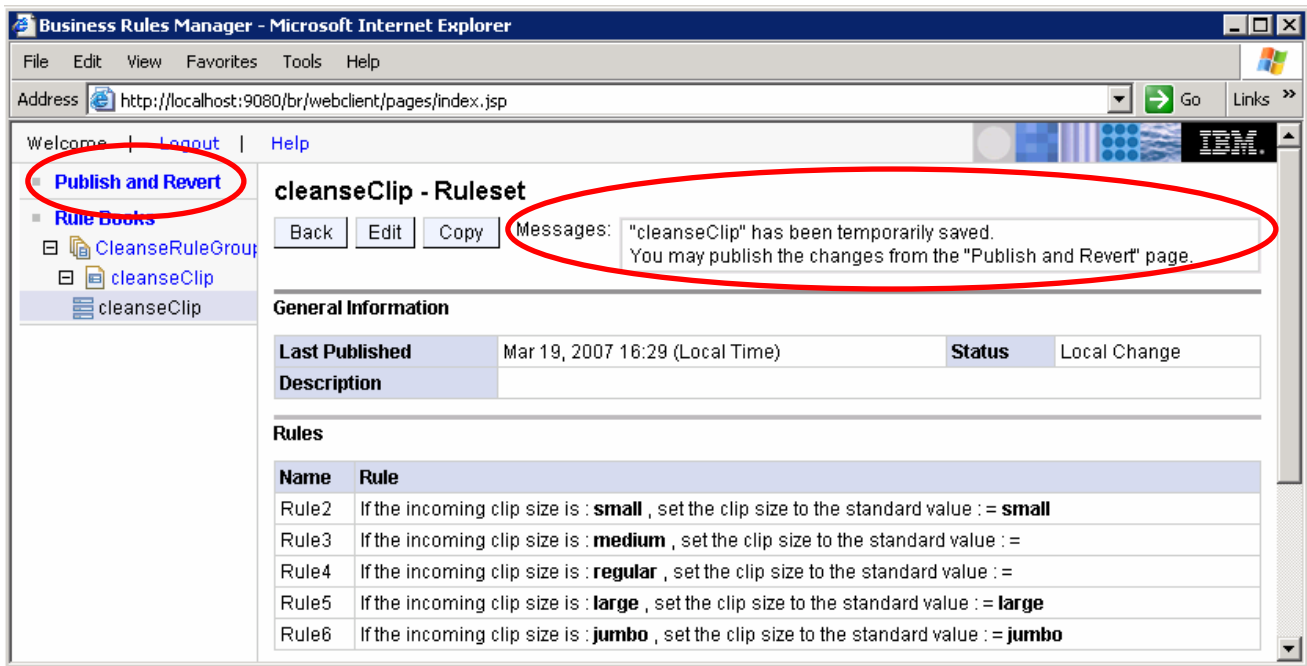
\_\_\_ 9. Edit **Rule6** with the clip size as Jumbo and change the output string from **large** to **jumbo**



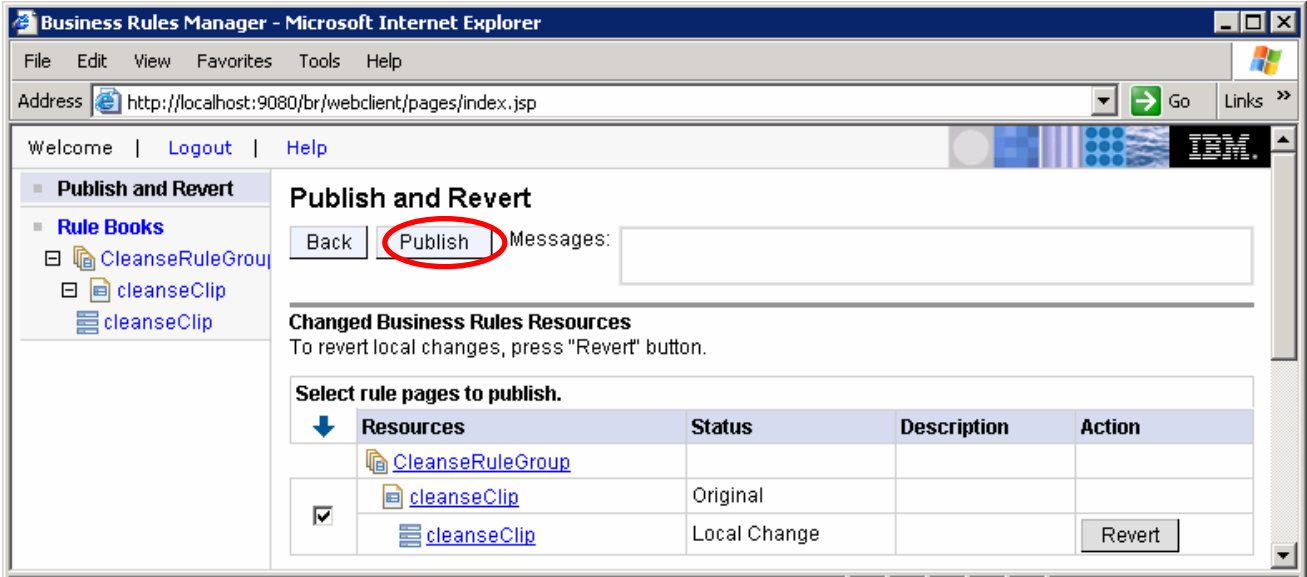


\_\_\_ 10. Click the **Save** button

\_\_\_ 11. Click the **Publish and Revert** link in the left navigation frame to publish the changes to the application server



\_\_\_ 12. Once here, you can either revert the changes you made or publish them. Click the **Publish** button.




\_\_\_ 13. The Business Rule Web tool test is complete

---

## Part 8: Test the business rule using component test

After wiring the components together, its time to test and see if the business rule works. In this section, you will learn to test a business integration module using the component test feature of WebSphere Integration Developer V6.0.2.

The Component test is a feature of the Eclipse-based WebSphere Rational® products from which WebSphere Integration Developer is built upon.

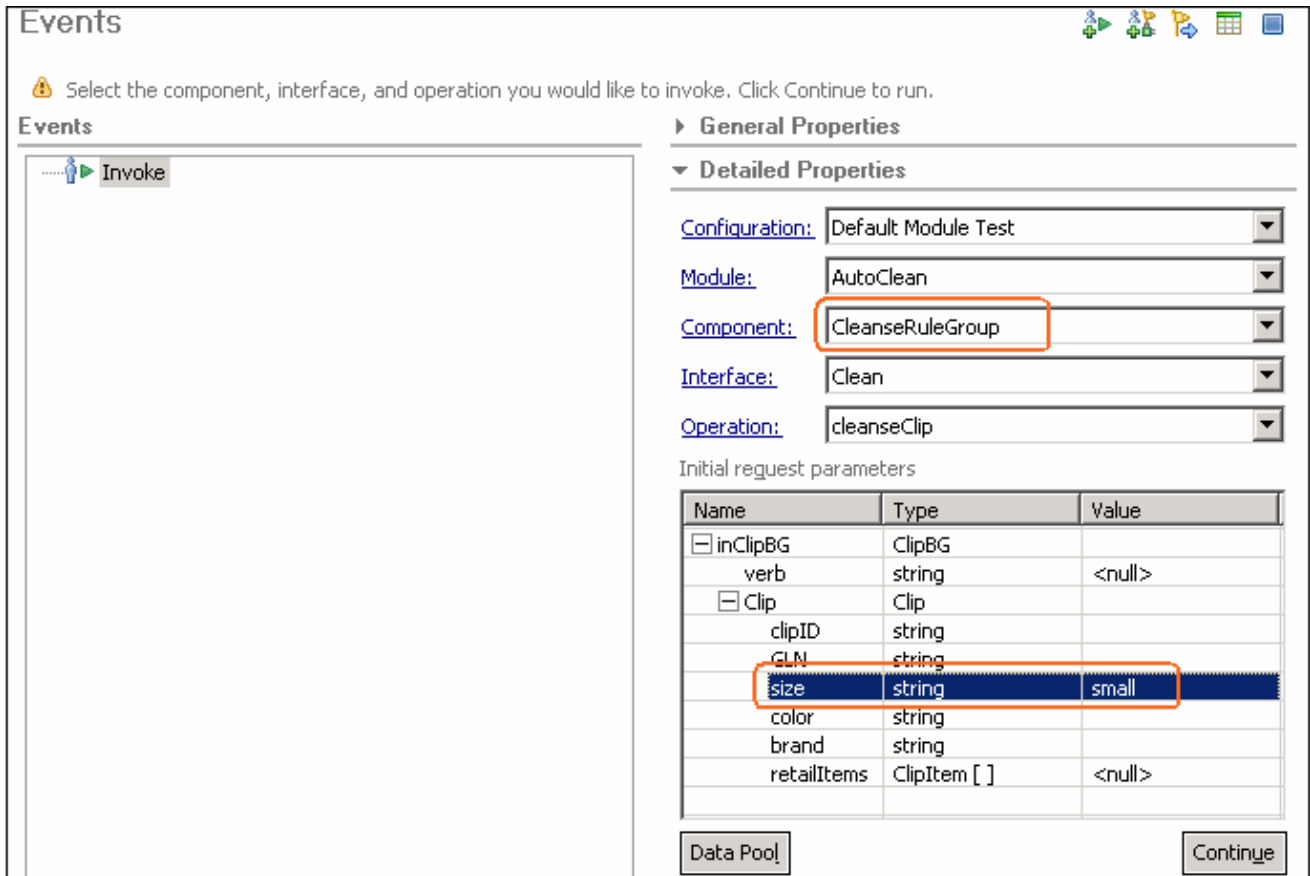
- \_\_\_ 1. Ensure the WebSphere Process Server (test server) is started
- \_\_\_ 2. Run the Component Test on the cleanseClip Business Rule
  - \_\_\_ a. In the Business Integration view, right click on the **AutoClean** module and select **Test → Test Module....** (  Test Module ) from the context menu

---

**Note:** You can also open the AutoClean Assembly Editor and right click on the component you would like to test and select Test Component.

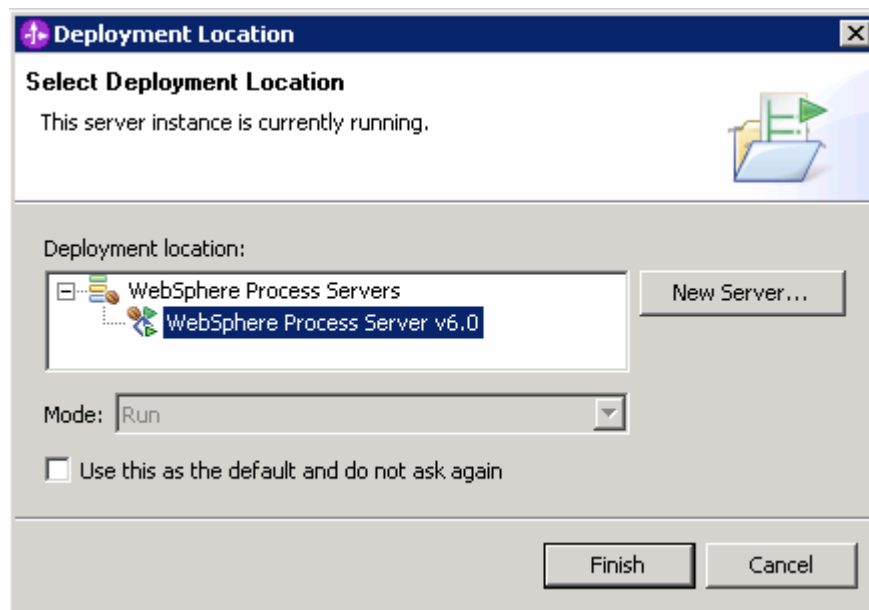
---

- \_\_\_ b. The Module tester is launched. Ensure the following information is provided under the Detailed Properties section of the Component Test editor
  - Configuration : **Default Module Test**
  - Module : **AutoClean**
  - Component : **CleansRuleGroup** ( Select from the drop down list)
  - Interface : **Clean**
  - Operation : **cleanseClip**
- \_\_\_ c. For the initial request parameters, enter the value for **size** as **small**. This will send the Clip size of “small” through the flow and should output “small”.



\_\_ d. Click the **Continue** button

\_\_ e. Select WebSphere Process Server 6.0 as the Deployment Location



\_\_ f. Click **Finish**

- \_\_\_ g. Your return parameter should be **small**. If you have a ? before small, that means the Business Rule did not fire. If you did not get a return, then Component Test did not run correctly, run again

**Events**

Invoke (CleanseRuleGroup:cleanseClip)

- Started
- Invoke (CleanseRuleGroup:cleanseClip)
- Return (CleanseRuleGroup:cleanseClip)**
- Stopped

**General Properties**

**Detailed Properties**

Module: [AutoClean](#)

Component: [CleanseRuleGroup](#)

Interface: [Clean](#)

Operation: [cleanseClip](#)

Return parameters:

Name	Type	Value
outClipBG	ClipBG	
verb	string	<unset>
Clip	Clip	
clipID	string	
GLN	string	
size	string	small
color	string	
brand	string	
retailItems	ClipItem [ ]	<null>

\_\_\_ 3. Stop and close the test component

- \_\_\_ a. Click the **Stop** (■) icon at the top right of the Events pane
- \_\_\_ b. Close the AutoClean\_Test component editor
- \_\_\_ c. Click **No** in the Save Resource dialog

\_\_\_ 4. Clean the WebSphere Process Server test environment

- \_\_\_ a. In the Servers view, right-click on WebSphere Process Server V6.0 and select **Add and remove projects....**
- \_\_\_ b. Select AutoCleanApp and click **< Remove**. Click **Finish**.
- \_\_\_ c. Right-click on **WebSphere Process Server V6.0** and select **Stop** from the context menu

## What you did in this exercise

In this lab you imported a Java implementation for running a selector/business rule scenario. You then created a rule group in order to create a business rule (rule set). Once the business rule implementation was set, you created a selector and finished wiring the example. When all was wired, you tested using the component test feature within WebSphere Integration Developer V6.0.2. To change the business rule templates, you installed and used the Business Rule Editor application.

## Solution instructions

- \_\_\_ 1. Follow the directions in the task “Initialize the Workspace for a Lab Exercise” from Part1, using the following values:

**<WORKSPACE>**

C:\Labfiles602\eXchange\AutoClean\workspace

**<PROJECT\_INTERCHANGE>**

C:\Labfiles602\eXchange\AutoClean\solution\AutoClean\_P1.zip

**<MODULE>**

n/a

**<DEPENDENT\_LIBRARIES>**

n/a

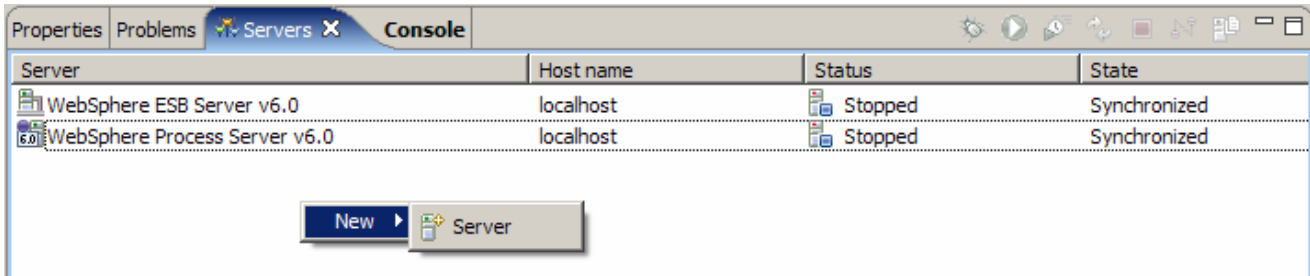
- \_\_\_ 2. Continue with **Part 7: Publish the AutoCleanApp**

# Task: Adding remote server to WebSphere Integration Developer test environment

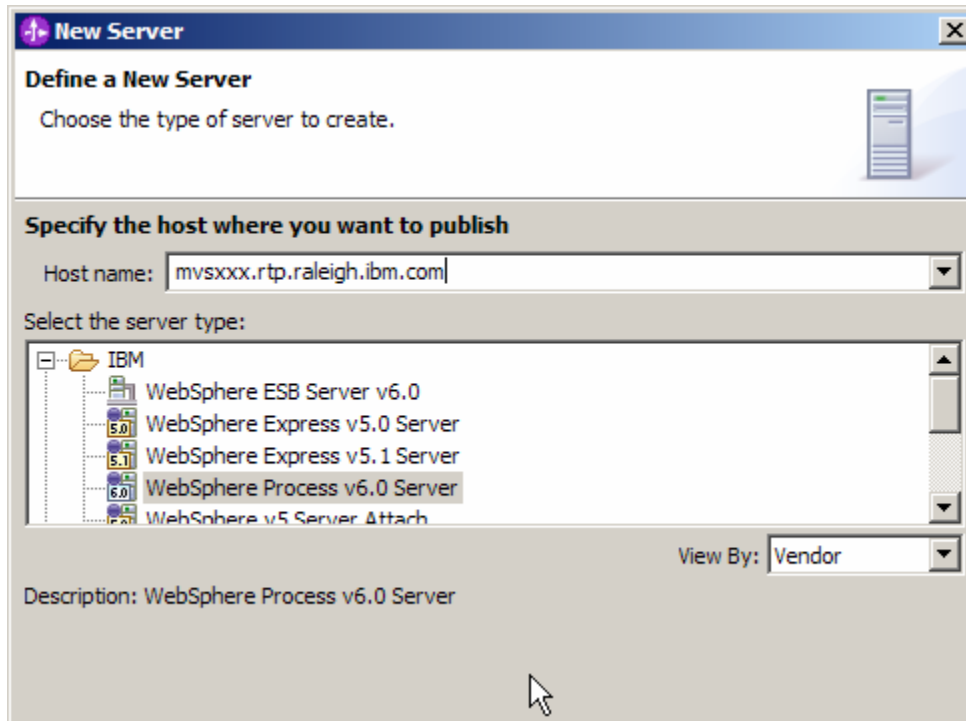
This task describes how to add a remote server to the WebSphere Integration Developer Test environment. The sample you will use is a z/OS machine.

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**.



- \_\_\_ 3. Specify host name to the remote server, **<HOSTNAME>**.
- \_\_\_ 4. Ensure that 'WebSphere Process V6.0 Server' is highlighted in the server type list.



- \_\_\_ 5. Click **Next**.



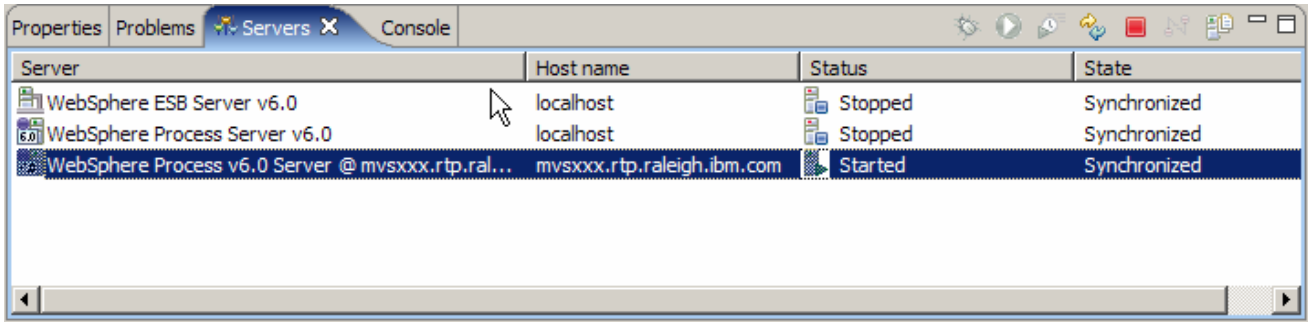
- \_\_\_ 6. On the WebSphere Server Settings page, select the radio button for **RMI** and change the ORB bootstrap port to the correct setting (<BOOTSTRAP\_PORT>).

The screenshot shows the 'New Server' dialog box with the following settings:

- WebSphere profile name: [Empty]
- 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: [Empty]
  - Password: [Empty]
- Server name: server1
- Server type:
  - BASE, Express or unmanaged Network Deployment server
  - Network Deployment server
  - Network Deployment server name: [Empty]
  - The server name is in the form of: <cell name>/<node name>/<server name>
  - For example, localhost/localhost/server1.
  - Detect button: Click this button to detect the server type.

Buttons at the bottom: < Back, Next >, Finish, Cancel

- \_\_\_ 7. Click **Finish**.
- \_\_\_ 8. The new server should be seen in the Server view.



\_\_\_ 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 c11sr01 open for e-business; process id is 0000012000000002
```