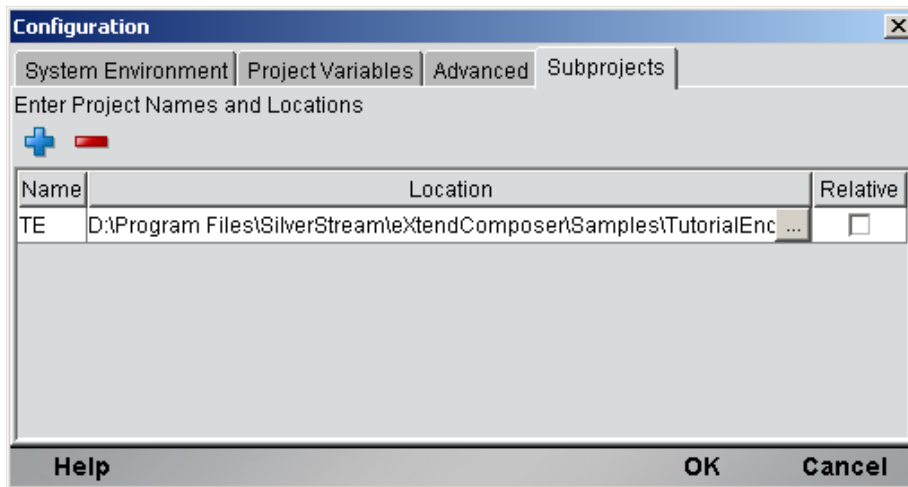

Process Manager Tutorial

This Tutorial assumes the user has already completed the regular Composer Tutorial, which uses the same resources and components that form the core of the tutorial that follows. Before proceeding, you should refamiliarize yourself with the Composer Tutorial. The procedures that follow are designed to acquaint you with the Process Manager rather than Composer *per se*.

Project Setup

➤ **Create a new Composer project: ProcessTutorialBegin**

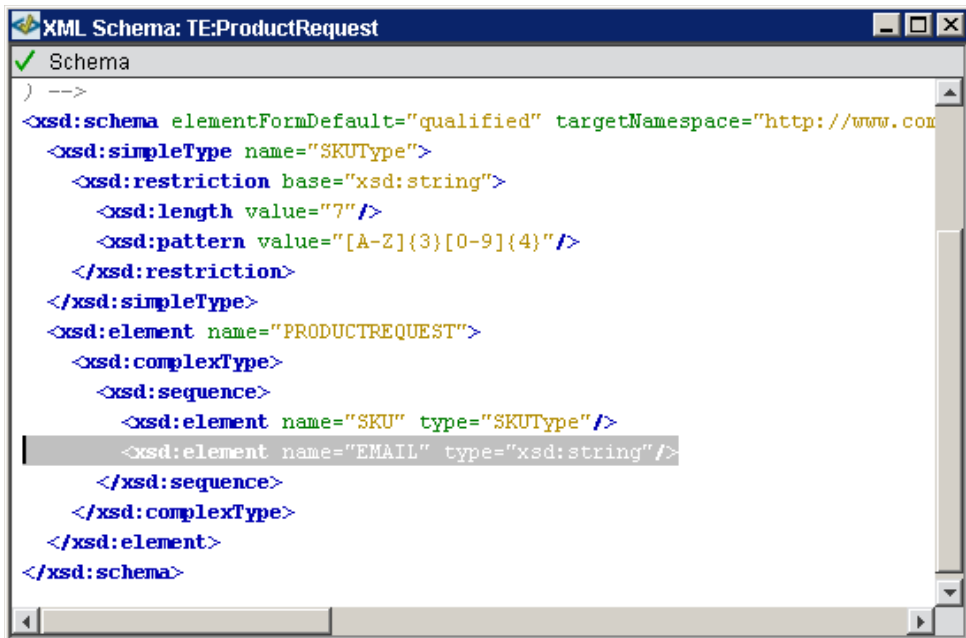
Add **TutorialEnd** as a subproject named **TE**. Create the subproject using the Subprojects tab of the Configuration dialog (**Tools > Configuration** in the Composer menubar).



XML Template Setup

Eventually, our In Stock reply activity will need to send an e-mail. We need to set this up ahead of time by making sure our ProductRequest schema specifies an EMAIL element.

➤ **Step 1: Add EMAIL to TE:ProductRequest Schema:**

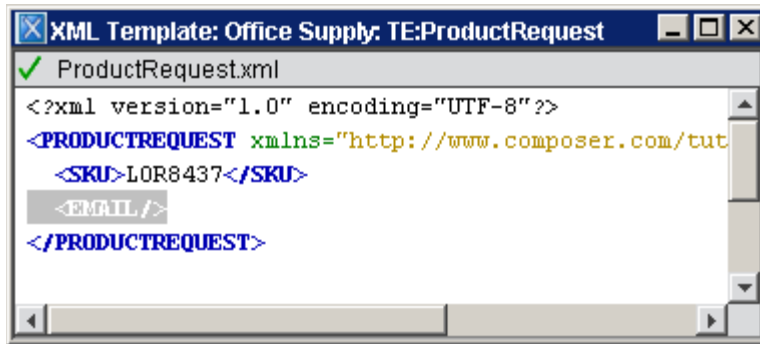


1. Open XML Schema **TE:ProductRequest**.
2. Switch to text view.
3. Add `<xsd:element name="EMAIL" type="xsd:string"/>` after `<xsd:element name="SKU" type="SKUType"/>` as shown by the highlighted text in the diagram above.
4. Save and close the schema.

➤ **Step 2: Add EMAIL to the sample document for XML Template TE:ProductRequest**

1. Open XML Template **TE:ProductRequest** in category **TE:Office Supply**
2. Switch to text view. (Right-click in content area and use the context menu to change views.)
3. Add `<EMAIL/>` after `<SKU>LOR8437</SKU>` as shown below.

4. Save and close the Template.



➤ **Create XML Template Category named "Office Supply Process"**

Add a Template called NotifyMeRequest in the new category.

Sample = `d:\program`

`files\silverstream\extendcomposer\samples\processtutorialfiles\notifymerequest.xml`

Process Setup

➤ **Create ProductInquiryLookup Process**

1. In Composer's File menu, choose **New xObject > Process**
2. Name = ProductInquiryLookup, Next
3. Input Template = TE:ProductRequest
4. Output Template = SystemAny.
5. Click **Finish**
6. From the menu bar, select **View > XML Documents > Show/Hide** and press **OK**

➤ **Create Composer Component Activity for ProductLookup**

1. Select a Composer Component from the Activity drop down button on the toolbar
2. Place the activity on the canvas
3. On the Properties Window select the Activity tab
4. Set Activity Name = ProductLookup
5. Set Component Type = JDBC
6. Set Component Name = TE:ProductLookup

➤ **Create Message Maps for the ProductLookup Activity**

1. Select the Messages tab
2. Click the plus (+) sign icon
3. For the Source Message select: ProcessInput
4. For the Source part type: Input (it may already be typed for you)
5. For the Target Message select: ProductLookupInput
6. For the Target part type: Input (it may already be typed for you)
7. Click **OK** to finish the Map

➤ **Create Composer Component Activity for InventoryLookup**

1. Select a Composer Component from the Activity drop down button on the toolbar
2. Place the activity on the canvas
3. On the Properties Window select the Activity tab
4. Set Activity Name = InventoryLookup
5. Set Component Type = JDBC
6. Set Component Name = TE:InventoryLookup

➤ **Create Message Maps for the InventoryLookup Activity**

1. Select the Messages tab
2. Click the plus (+) sign icon
3. For the Source Message select: ProcessInput
4. For the Source part type: Input (it may already be typed for you)
5. For the Target Message select: InventoryLookupInput
6. For the Target part type: Input (it may already be typed for you)
7. Click **OK** to finish the Map

➤ **Create Composer Component Activity for MergeProductAndInventory**

1. Select a Composer Component from the Activity drop down button on the toolbar
2. Place the activity on the canvas
3. On the Properties Window select the Activity tab
4. Set Activity Name = MergeProductAndInventory
5. Set Component Type = XML Map
6. Set Component Name = TE:MergeProductAndInventory

➤ **Create Control Links between the Activities**

1. Click the Links button on the toolbar
2. Draw a link from InventoryLookup to MergeProductAndInventory
3. Click the Links button on the toolbar again
4. Draw a link from ProductLookup to MergeProductAndInventory

➤ **Create Message Maps for the MergeProductAndInventory Activity**

1. Select the Messages tab
2. Click the plus (+) icon
3. For the **Source Message** select: ProductLookupOutput
4. For the **Source** part type: Output (it may already be typed for you)
5. For the **Target Message** select: MergeProductAndInventory
6. For the **Target** part type: Input (it may already be typed for you)
7. Click **OK** to finish the Map
8. Click the plus (+) icon again
9. For the **Source Message** select: InventoryLookupOutput.
10. For the **Source** part type: Output (it may already be typed for you)
11. For the **Target Message** select: MergeProductAndInventory
12. For the **Target** part type: Input1
13. Click **OK** to finish the Map

➤ **Create the ProcessOutput Message**

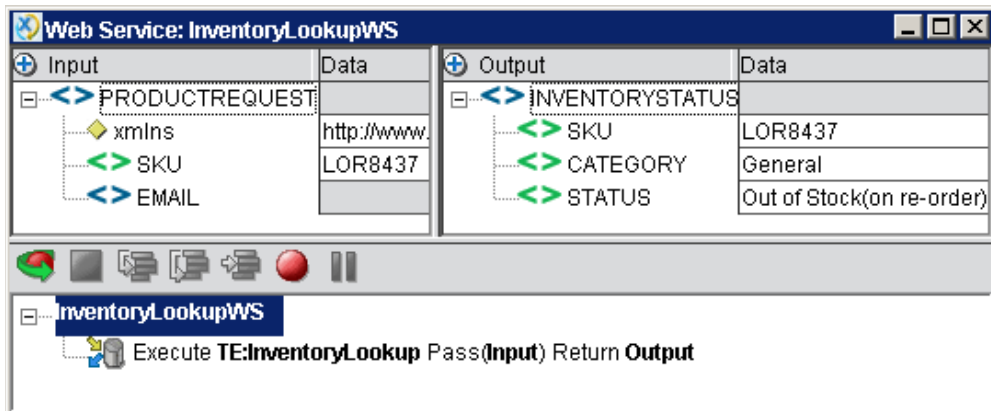
1. Click the canvas background
2. If it is not already selected click the Messages tab
3. Click the plus (+) (+) icon
4. For the **Source Message**, select: MergeProductAndInventory
5. For the **Source** part type: Output (it may already be typed for you)
6. For the **Target Message** select: ProcessOutput
7. For the **Target** part type: Output
8. Click **OK** to finish the Map
9. Test by pressing the **Execute** button
10. Press **OK** when done

InventoryLookup as an External Web Service

This step is provided to show how an external Web Service that provides an Inventory lookup can be incorporated into the project. Normally the external Web Service would be provided by a third party or sister organization but for demonstration purposes, it and the WSDL for it are created within the current project.

➤ Create a new Web Service xObject named InventoryLookupWS that calls the component TE:InventoryLookup

1. In Composer's File menu, choose **New xObject > Service > WebService**
1. Input Message = TE:Office Supply ProductRequest
2. Output Message = TE:Office Supply InventoryStatus
3. Component Action to TE:InventoryLookup
4. Test, Save and close.



➤ Create WSDL for the Web Service

1. **File > New xObject > Resource > WSDL**
2. Name = InventoryLookupWS
3. Associate Web Service = checked
4. Service = InventoryLookupWS
5. Generate SOAP Binding = checked
6. Binding Style = Document
7. URL = http://localhost:80/XCTutorial/ProcessTutorialBegin/inventorylookupws

8. Click Finish and then **OK** to dialog mentioning No Schemas

➤ **Deploy the Inventory Web Service**

1. **File > Deploy**
2. Panel 1
3. Server Type = Silverstream J2EE
4. JAR Filename = ProcessTutorialBegin.jar
5. Deployment Context = com.composer.processtutorial
6. Click **Next** 3 times
7. SOAP HTTP based Service Triggers panel
8. Add
Service = **InventoryLookupWS**
9. Next 3 times to Upload panel
10. Upload Panel
11. Select 'Yes'
12. Server Name / Port = localhost:80
13. Server Database Name = XCTutorial
14. User Name = <user name>
15. Password = <password>
16. Finish

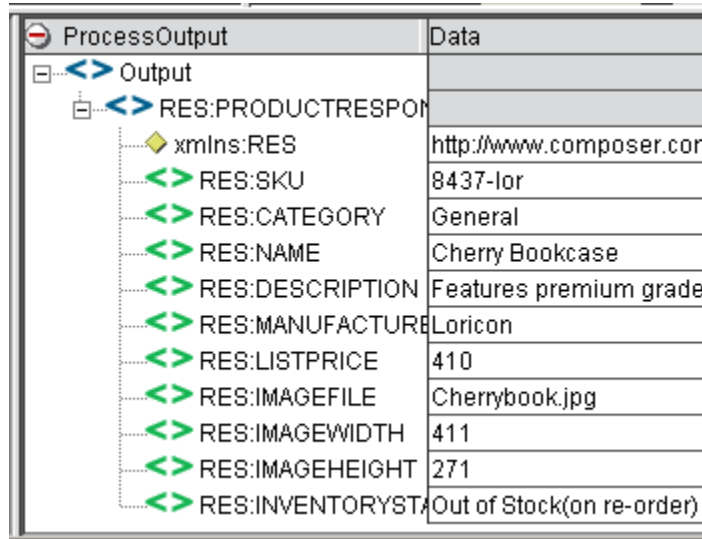
➤ **Modify the ProductInquiryLookup process to use the deployed Web Service**

1. Select the InventoryLookup activity
2. Select the Activity tab in the Properties window
3. Select 'Web Service Send' for Activity Type
4. Select InventoryLookupWS for WSDL Resource

➤ **Test the new version of the ProductInquiryLookup process**

1. Press the Animate button
2. Step Over both the ProductLookup and the InventoryLookup activities
3. Step Into the MergeProductAndInventory activity
4. Press Run to Breakpoint/End

5. The ProcessOutput message should look like the picture below



ProcessOutput	Data
Output	
RES:PRODUCTRESPON	
xmlns:RES	http://www.composer.cor
RES:SKU	8437-lor
RES:CATEGORY	General
RES:NAME	Cherry Bookcase
RES:DESCRIPTION	Features premium grade
RES:MANUFACTURE	Loricon
RES:LISTPRICE	410
RES:IMAGEFILE	Cherrybook.jpg
RES:IMAGEWIDTH	411
RES:IMAGEHEIGHT	271
RES:INVENTORYSTA	Out of Stock(on re-order)

6. **Save** and close the process.

➤ **Create ProductInquiry Process**

1. File > New xObject > Process
2. Name = ProductInquiry, Next
3. Input Template = TE:ProductRequest
4. Output Template = SystemAny, Finish
5. View > XML Documents > Show/Hide, **OK**

➤ **Create SubProcess Activity**

1. Select a Subprocess from the Activity drop down button on the toolbar
2. Place the activity on the canvas
3. On the Properties Window select the Activity tab
4. Set Activity Name = ProductInquiryLookup
5. Set Process Name = ProductInquiryLookup
6. Set Timeout = 600

7. Set Retry Count = 2
8. Set Retry Interval = 2

➤ **Create Message Maps for the ProductInquiryLookup Subprocess**

1. Select Messages tab
2. Click the plus (+) icon
3. For the Source Message select: ProcessInput
4. For the Source part type: Input (it may already be typed for you)
5. For the Target Message select: ProductLookupInput
6. For the Target part type: Input (it may already be typed for you)
7. Click **OK** to finish the Map

➤ **Create a Component for an In Stock reply activity**

(This component will simply send an e-mail back to the requestor, but could easily be a whole other set of activities or processes to allow the requestor to order an item, perform another inquiry, etc. This component will take two input parts. The first part is the results of the product inquiry. These will be referenced in the e-mail that is sent. The second part contains the original request that contains the e-mail address to respond to.)

1. Create a new XML Map component named "InStockReply"
2. For the Input Message Input part, select TE:Office Supply as the template category and ProductResponse as the template name
3. For the Input Message, add a second part Input1
4. For the Input1 part, select TE:Office Supply as the template category and ProductRequest as the template name
5. Create a Send Mail action
6. Mail Recipient = Input1.XPath("PRODUCTREQUEST/EMAIL")
7. Mail Sender = "pm@composer.com"
8. Mail Subject = "Product Information for " + Input.XPath("PRODUCTREQUEST/SKU")
9. Mail Body = "Product: " + Input.XPath("RES:PRODUCTRESPONSE/RES:SKU") +
"\r\n" +
"Description: " +
Input.XPath("RES:PRODUCTRESPONSE/RES:DESCRIPTION") + "\r\n" +
"Stock Status: " +
Input.XPath("RES:PRODUCTRESPONSE/RES:INVENTORYSTATUS") +
"\r\n\r\n" +
"Thank you for your inquiry!"

```
Mail Server = "your.mail.server"
```

10. Save and Close the component

➤ Create an Activity for the In Stock reply

1. Return to the ProductInquiry process
2. Select a Composer Component from the Activity drop down button on the toolbar
3. Place the activity on the canvas
4. On the Properties Window select the Activity tab
5. Set Activity Name = InStockReply
6. Set Component Type = XML Map
7. Set Component Name = InStockReply

➤ Create a Control Link for the Activities

1. Click the Links button on the toolbar
2. Draw a link from ProductInquiryLookup to InStockReply

➤ Create Message Maps for the InStockReply activity

1. Select Messages tab for InStockReply
2. Click the plus (+) icon
3. For the Source Message select: ProductInquiryLookupOutput
4. For the Source part type: Output (it may already be typed for you)
5. For the Target Message select: InStockReplyInput
6. For the Target part type: Input (it may already be typed for you)
7. Click **OK** to finish the Map
8. Click the plus (+) icon again
9. For the Source Message select: ProcessInput
10. For the Source part type: Input (it may already be typed for you)
11. For the Target Message select: InStockReplyInput
12. For the Target part type: Input1
13. Click **OK** to finish the Map

➤ Test the process

1. Enter "DAD7777" for SKU in the Input message
2. Enter your e-mail address for EMAIL

3. Press the execute button you should receive an e-mail.
4. Click the Reset icon:



Reset icon

➤ Add a Condition to the Control Link

Ignoring errors for the moment, we anticipate two types of responses from the Inquiry: In Stock or Out of Stock. Each response will be handled by a separate activity. To ensure the In Stock activity only executes when items are in stock we add a condition to the link leading into the activity. As its currently defined, the link will be traversed and the activity executed provided the Lookup subprocess produces a normal result message named "ProductInquiryLookupOutput."

1. Select the Control Link between ProductInquiryLookup and InStockReply
2. On the Properties window select the Link tab
3. Click the Expression icon in the Value cell for Condition
4. Enter the following:

```
Output/RES:PRODUCTRESPONSE/RES:INVENTORYSTATUS = "In Stock"
```

NOTE This condition will ensure that the link is traversed and the InStockReply activity executed only if the INVENTORYSTATUS element in the message contains the value "In Stock".

➤ Test the process

1. Enter "DAD7777" for SKU in the Input message
2. Enter your e-mail address for EMAIL
3. Press the execute button you should receive an e-mail.
4. Test again by entering "LOR8437" instead of "DAD7777" you should NOT get an e-mail.

➤ Create a Component for an Out of Stock reply activity

1. Copy the XML Map component named InStockReply
2. Paste it into the category and rename it to *OutOfStockReply*
3. Add another part to its Input Message (this will contain identify information about the process)
4. Open *OutOfStockReply*
5. Make the new part visible in the component editor (**View > XML Documents > Show/Hide**)

➤ Edit the Send Mail action

1. Change the Mail Body to:

```
"Product: " + Input.XPath("RES:PRODUCTRESPONSE/RES:SKU") + "\r\n" +  
"Description: " + Input.XPath("RES:PRODUCTRESPONSE/RES:DESCRIPTION") +  
"\r\n" +  
"Stock Status: "+Input.XPath("RES:PRODUCTRESPONSE/RES:INVENTORYSTATUS") +  
"Please click this link if you'd like to be notified when the item is in  
stock." + "\r\n\r\n" +  
"http://localhost/XCTutorial/ProcessTutorialBegin/notifyme?InquiryID=" +  
Input2.XPath("ProcessInfo/ProcessID") +  
&EMAIL=" + Input1.XPath("PRODUCTREQUEST/EMAIL") + "\r\n\r\n" +  
"Thank you for your inquiry!"
```

NOTE The URL constructed in the Mail Body contains an InquiryID parameter. The InquiryID is the unique ProcessID assigned to the process initiated by the users original inquiry. This ID will allow the unfinished process to be recalled out of persistent storage to continue processing the user's request and be notified when an item is in stock. This will be more apparent when you create the NotifyMe activity below.

2. **Save** and **Close** the component

➤ Create an Activity for the Out Of Stock reply

1. Return to the ProductInquiry process
2. Select a Composer Component from the Activity drop down button on the toolbar
3. Place the activity on the canvas
4. On the Properties Window select the Activity tab
5. Set Activity Name = OutOfStockReply
6. Set Component Type = XML Map
7. Set Component Name = OutOfStockReply

➤ Create a Control Link for the activities

1. Click the Links button on the toolbar
2. Draw a link from *ProductInquiryLookup* to *OutOfStockReply*
3. Next, in the Properties window, select the Link tab
4. Click the XPath expression icon in the Condition field to add a condition to the link
5. Type the following expression to follow this link when an item is out of stock:

```
Output/RES:PRODUCTRESPONSE/RES:INVENTORYSTATUS =  
"Out of Stock(on re-order)"
```

6. Press **OK** to close the expression builder

➤ **Create Message Maps for the OutOfStockReply activity**

1. Select the OutOfStockReply activity
2. Select the Messages tab in the Properties windows for OutOfStockReply
3. Click the plus (+) icon
4. For the Source Message select: ProductInquiryLookupOutput
5. For the Source part type: Output (it may already be typed for you)
6. For the Target Message select: OutOfStockReplyInput
7. For the Target part type: Input (it may already be typed for you)
8. Click **OK** to finish the Map
9. Click the plus (+) icon again
10. For the Source Message select: ProcessInput
11. For the Source part type: Input (it may already be typed for you)
12. For the Target Message select: OutOfStockReplyInput
13. For the Target part type: Input1
14. Click **OK** to finish the Map
15. Click the plus (+) icon again
16. For the Source Message select: ProcessInput
17. For the Source part type: Process
18. For the Target Message select: OutOfStockReplyInput
19. For the Target part type: Input2
20. Click **OK** to finish the Map

➤ **Test the Process**

1. Enter "LOR84437" for SKU in the Input message
2. Enter your e-mail address for EMAIL
3. Press the execute button you should receive an e-mail offering a link for you to click.

➤ **Test the process**

1. Enter "DAD7777" for SKU in the Input message
2. Enter your e-mail address for EMAIL
3. Press the execute button you should receive an e-mail

4. Test again by entering "LOR8437" instead of "DAD7777"; you should *not* get an e-mail.

➤ **Create a Web Service Receive Activity that Waits for a NotifyMe Request**

When this Web Service Receive activity executes, the process will essentially halt and be saved off into persistence (i.e. the process database). The process and activity remain in this state until the timeout period passes or the user clicks the URL sent in the e-mail. The URL is attached to a Web Service that will be the implementation for this activity. The implementation is then responsible for re-activating the process so it can complete.

1. Return to the ProductInquiry process
2. Select a Web Service Receive from the Activity drop down button on the toolbar
3. Place the activity on the canvas
4. On the Properties Window select the Activity tab
5. Set Activity Name = NotifyMe
6. Set Implementation Type = Web Service

➤ **Create a Control Link for NotifyMe**

1. Click the Links button on the toolbar
2. Draw a link from OutOfStockReply to NotifyMe

➤ **Create Message Maps for the NotifyMe activity**

1. Select the Messages tab in the Properties windows for NotifyMe
2. Click the plus (+) icon
3. For the Source Message select: ProductInquiryLookupOutput
4. For the Source part type: Output (it may already be typed for you)
5. For the Target Message select: NotifyMeInput
6. For the Target part type: Input (it may already be typed for you)
7. Click **OK** to finish the Map

➤ **Create a Web Service as the Implementation for the NotifyMe activity**

Up to this point you have been creating implementations for your processes and layering activities on top of them. At this point you just created an activity and will now create its implementation. Composer supports both bottom up or top down project design and creation.

1. Create a new Web Service named NotifyMe
2. Define the Input Message with the following templates:
Input: Category = OfficeSupplyProcess Template = NotifyMeRequest

Input1: Category = Process Template = PendingActivity

Input2: Category = TE:OfficeSupply Template = ProductResponse

3. Define the Output Message with the following template:

Output: Category = System Template = Any

➤ Create a Find Waiting Activity action

1. From the menu bar: **Action > New Action > Process > Find Waiting Activity**, select the 'Find' tab
2. Click 'Process Name and CorrelationID. (This means we will find the 'waiting activity' in the process database by a unique combination of a Process Name and a CorrelationID of our choosing.)
3. Select 'ProductInquiry' as the Process Name
4. For the CorrelationID select XPath 'Input' and enter 'NotifyMeRequest/InquiryID' as the XPath expression

NOTE This is why we coded the InquiryID into the URL in the e-mail sent to the user so we would receive it back as a method of finding the process handling the user's original inquiry.

5. Select the 'Message' tab
6. In the first row under 'Parts' type Input for the Source Part

NOTE This will retrieve the waiting activity's Input message part into this component when the activity is found
7. In the first row under 'Parts' select Input2 for the target part

NOTE This is the part in this component that will receive the waiting activity's Input message part
8. In the field labeled 'Target Part for Waiting Activity's Process Information' enter Input1

NOTE This will contain the information necessary to release the waiting activity/process
9. Press **OK** to close the dialog.

➤ Create the Output message for the NotifyMe activity

1. Create a Map action with source part = Input2 and source XPath = RES:PRODUCTRESPONSE, and target part = Output and target XPath = NOTIFYWHENINSTOCK
2. Create another Map action with source part = Input and source XPath = NotifyMeRequest/EMAIL, and target part = Output and target XPath = NOTIFYWHENINSTOCK/EMAIL

NOTE This creates an Output message that combines the EMAIL address to notify with the Inquiry information. In the tutorial scenario, this could then be sent on to another activity or process to handle the notification.

➤ Create a Release Waiting Activity action

1. From the menu bar: Action > New Action > Process > Release Waiting Activity
2. In field labeled 'Part containing the Waiting activity's Process information' enter: Input1
NOTE The PendingActivity document placed in Input1 by the Find Waiting Activity action has all the information needed to release a waiting activity.
3. In the first row under 'Parts' select Output for the Source Part
NOTE This identifies which part in this component will become the output message for the waiting activity
4. In the first row under 'Parts' type Output for the target part
NOTE This identifies the part in the waiting activity that will receive the output message
5. Complete the Web Service Receive activity NotifyMe
6. Select the NotifyMe activity
7. Set the Web Service Name = NotifyMe
8. In the CorrelationID field enter: \$ProcessInput/Process/ProcessInfo/ProcessID
NOTE Each process is assigned a unique ID when it is instantiated. This ID is available in the ProcessInput message of every process. For simplicity we will use the ProcessID to tag this Web Service Receive activity before it enters its waiting state. This is the same ID embedded in the URL of the e-mail. That URL invokes the activity's implementation which uses the ID to Find and Release the waiting activity.
9. In the Timeout field enter: 600
NOTE All Web Service Receive activity's should be assigned a Timeout. This activity may never get executed if the user doesn't respond to the e-mail sent in OutOfStockReply. For this reason it will have a timeout period assigned which if exceeded will allow the process to terminate normally.
10. Save the process

➤ Create an Activity for the Notification

1. Select a Composer Component from the Activity drop down button on the toolbar
2. Place the activity on the canvas
3. On the Properties Window select the Activity tab
4. Set Activity Name = Notification

➤ **Create a Control Link for Notification**

1. Click the **Links** button on the toolbar
2. Draw a link from NotifyMe to Notification

➤ **Create Message Maps for the Notification activity**

1. Select the Messages tab in the Properties windows for Notification
2. Click the plus (+) icon
3. For the Source Message select: NotifyMeOutput
4. For the Source part type: Output (it may already be typed for you)
5. For the Target Message select: NotificationInput
6. For the Target part type: Input (it may already be typed for you)
7. Click **OK** to finish the Map

➤ **Create a Composer Component as the Implementation for the Notification activity**

1. Create a new XML Map component named Notification
2. Define the Input Message with the following templates:
3. Input: Category = System Template = Any
4. Define the Output Message with the following template:
5. Output: Category = System Template = Any
6. Create a **Send Mail** action

NOTE Under normal circumstances, this implementation for the Notification activity would most likely spawn a separate process to periodically check inventory and then notify the business partner when the item was back in stock. For the tutorial we will simply send another e-mail.

1. Mail Recipient = `Input.XPath("NOTIFYWHENINSTOCK/EMAIL")`
2. Mail Sender = `"pm@composer.com"`
3. Mail Subject = `"The Product " + Input.XPath("NOTIFYWHENINSTOCK /SKU") + " is now in stock."`
4. Mail Body = `"The Product " + Input.XPath("NOTIFYWHENINSTOCK /SKU") + " is now in stock." + "\r\n\r\n" + "Thank you for your inquiry!"`
5. Mail Server = `"your.mail.server"`

6. Save and exit the component

NOTE For demonstration purposes the tutorial does no further processing. In a real application, the implementation for the Notification activity might be a separate Web Service or perhaps another subprocess.

➤ **Complete the Notification activity**

1. Return to the ProductInquiry process
2. Select the Notification activity
3. Set Component Type = XML Map
4. Set Component Name = Notification
5. Save the process

➤ **Test End-to-End Animation**

1. Click the animation **Start** button or press **F5**
2. Click the **Step Into** button or press **F7**; the ProductInquiryLookup subprocess should open
3. Click the ProductLookup activity
4. Click the **Step Into** button again; the ProductLookup component should open
5. Click the **Run to Breakpoint/End** button or press **F9**. You should return to ProductInquiryLookup with InventoryLookup selected
6. Click the **Step Over** button or press **F8**; the MergeProductAndInventory activity should be selected
7. Click the **Step Over** button and you should return to ProductInquiry with OutOfStockReply selected
8. Step Over OutOfStockReply and you should receive an e-mail shortly
9. Open the mail when it arrives and write down the InquiryID you see in the URL
NOTE Since the URL is only active after deploying to an application server, an alternate technique is used for testing during design time end to end testing.
10. NotifyMe should be selected
11. Step Into NotifyMe and the component should open with its samples displayed
12. Double click InquiryID in the Input part, enter the Inquiry ID you wrote down, and press **OK**
13. Click the **Animate** button or press **F5**
14. Click Step Into twice, the Find Waiting Activity should execute placing a PendingActivity document in the Input1 part
15. Click Step Into twice again and the Output message for NotifyMe should be created

16. Click Step Into again and the animation will end
17. From the menu bar select **Window > ProductInquiry**
18. You should see that NotifyMe is completed and Notification is selected
19. Click the Run to Breakpoint/End button and the process completes
20. Close the process

➤ **Create a Web Service wrapper for the ProductInquiry process**

1. From the menu bar select File > New xObject > Service > Web Service
2. For the Name field enter: ProductInquiryWS and press Next
3. For the Input Message category for the Input part select: TE:Office Supply
4. For the Template Name select: ProductRequest
5. Press the Finish button

➤ **Create a Process Execute action**

1. From the menu bar select: **Action > New Action > Process > Process Execute**
2. For the Execute Method control select: Spawn
3. For the Process Component control select: ProductInquiry
4. Save and close the Web Service

➤ **Create WSDL for the ProductInquiryWS Web Service**

1. From the menu bar select: **File > New xObject > Resource > WSDL**
2. For the Name field enter: ProductInquiry
3. Press the Next button and the Associate Web Service panel appears
4. Make sure the Associate Web Service check box is checked
5. For the Service select: ProductInquiryWS
6. Make sure Generate SOAP Binding is checked
7. Make sure the Binding Style is **Document**
8. For the URL enter:
http://localhost:80/XCTutorial/ProcessTutorialBegin/productinquiryws
9. Press the Finish button and press **OK** when the 'No schemas are used...' dialog appears
10. Save and close the WSDL

➤ Deploy the project

1. From the menu bar select **File > Deploy**
2. Make sure the values for each field are as follows:
Silverstream J2EE
D:\Program Files\Silverstream\ExtendComposer\Samples\ProcessTutorialBegin\staging
ProcessTutorialBegin.jar
com.composer.processtutorial
3. Press the Next button and the Servlet based Service Triggers panel appears
4. Press the Add button and a blank row appears
NOTE The user will indicate that they want to be notified by clicking a URL encoded with parameters inside their e-mail. So the notifyMe Web Service will be deployed as a Servlet taking HTTP Parameters as input.
5. For the Service select: NotifyMe
6. For the Servlet Type select: Params(URL/Form)
7. For the Output Type select: XML
8. Leave Role blank
9. For the URL Path enter: notifyme
10. Press the Next button two times and the SOAP HTTP based Service Triggers panel appears
11. Press the Add button and a blank row appears
12. For Service select: ProductInquiryWS, the remaining fields on the row should fill automatically
13. Press the Next button three times until the Upload panel appears
14. Enter appropriate values for each field such as:
localhost:80
XCTutorial
<name>
<pw>
15. Press the **Finish** button
16. Press **Yes** if the "Jar File exists" dialog appears

➤ Test the Deployed ProductInquiry process (Option 1)

1. Create an XML Map Component named ProductInquiryClient

NOTE This component will simulate a business partner contacting the ProductInquiry process using WSDL generated for the Web Service wrapping the process.

2. Define the Input Message with the following templates:
3. Input: Category = TE:OfficeSupply Template = ProductRequest
4. Define the Output Message with the following template:
5. Output: Category = TE:OfficeSupply Template = ProductResponse

➤ **Create a Web Service Interchange action**

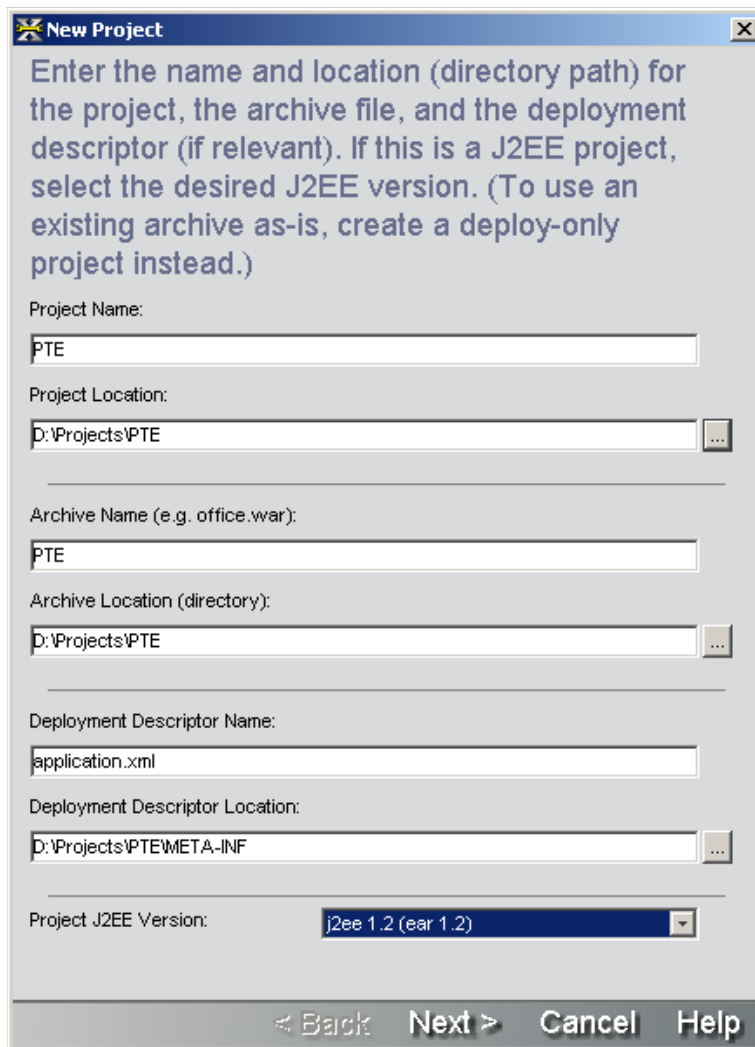
1. For the WSDL Resource select: ProductInquiry
2. Select the "Messages" tab
3. For the Input part expression enter: Input.XPath("PRODUCTREQUEST")
4. For the Output part expression enter: Output
5. Close the action
6. Edit the EMAIL element in the Input part for the component by entering your e-mail address
7. Save the component

➤ **Test the ProductInquiry client**

1. Press the **Execute** button
2. You should see a ProcessInfo receipt in the Output part
3. You should get an e-mail
4. Click the link in the e-mail
5. You should receive a second e-mail

➤ **Test the Deployed ProductInquiry process (Option 2)**

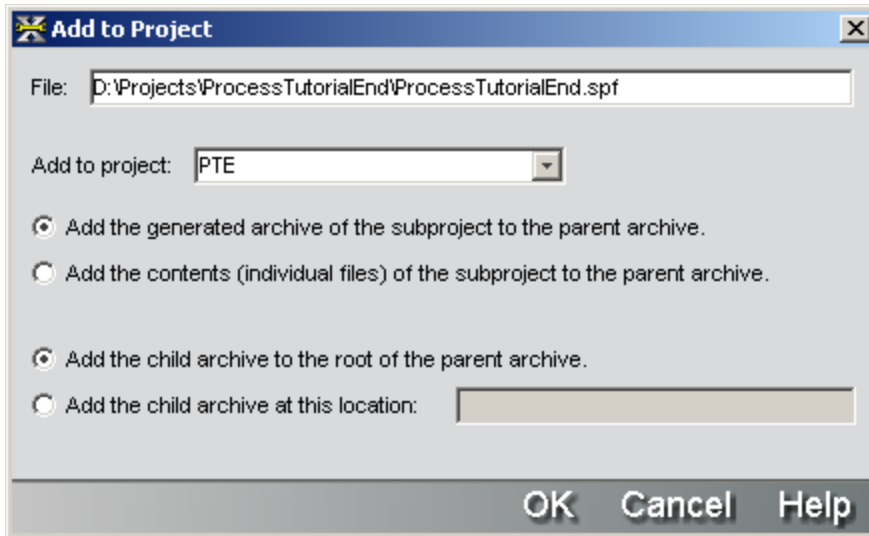
1. Start the eXtend Workbench
2. Create a new EAR project



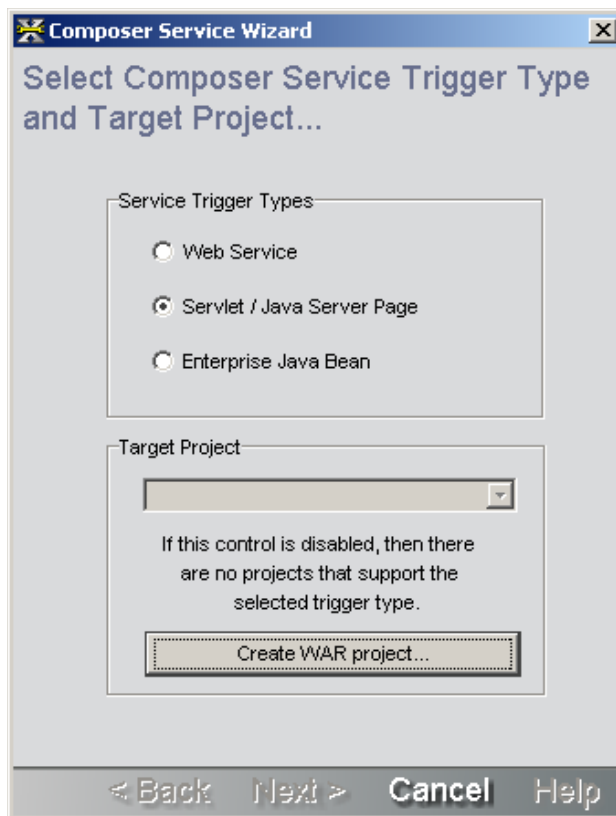
3. For Project Name enter: PTE
4. For Project Location enter the appropriate path
5. Make sure to select **j2ee 1.2** (ear1.2)
6. Answer Yes to create the new directory
7. Press the Finish button

➤ **Add ProcessTutorialEnd as a subproject to PTE**

1. Select the EAR file `spf` and press the RMB (right mouse button)
2. Select Add Subproject to project and File Chooser appears
3. Navigate to **ProcessTutorialEnd.spf** and click Open and Add to Project dialog appears

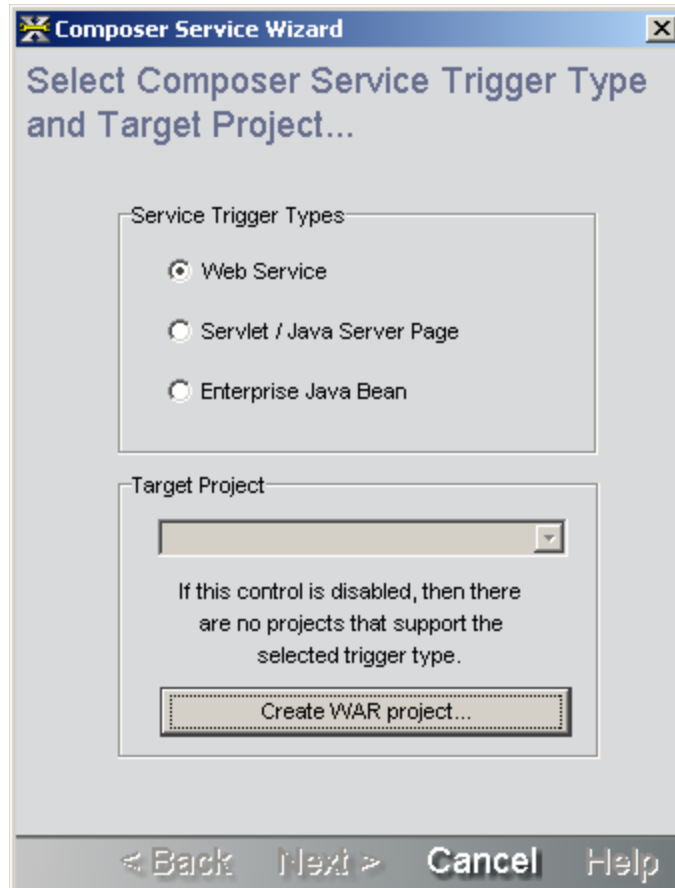


4. Click **OK**.

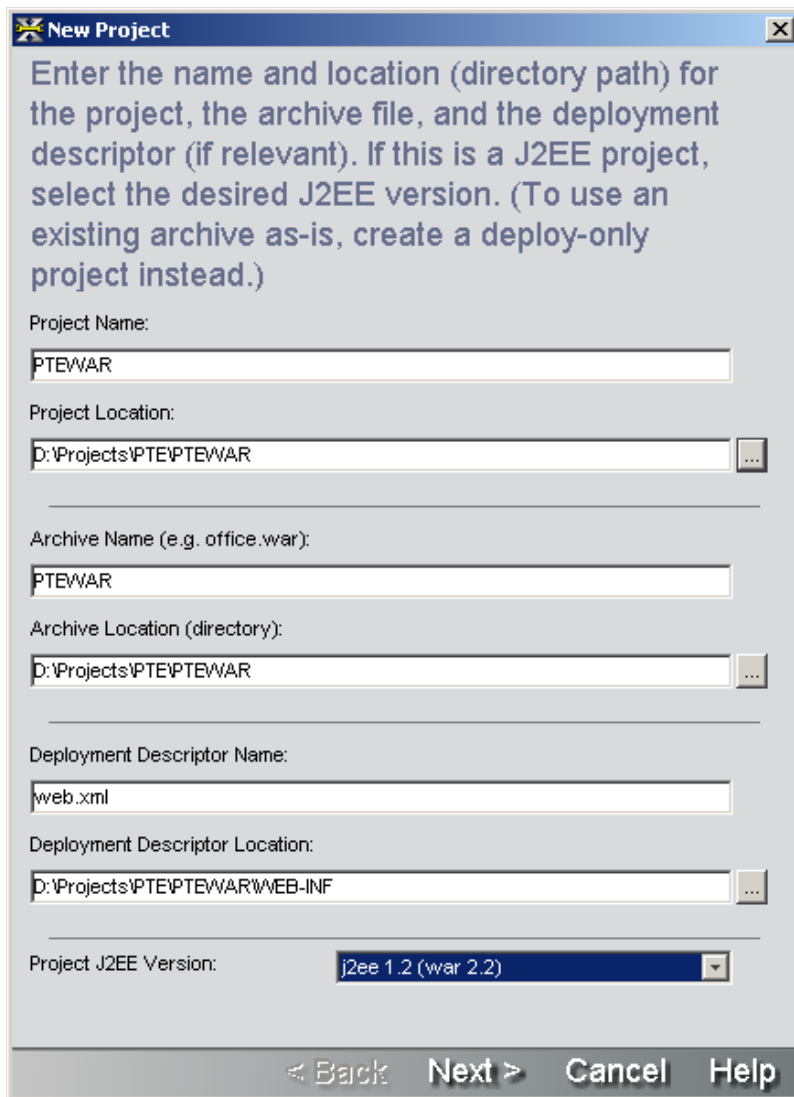


Create WAR project

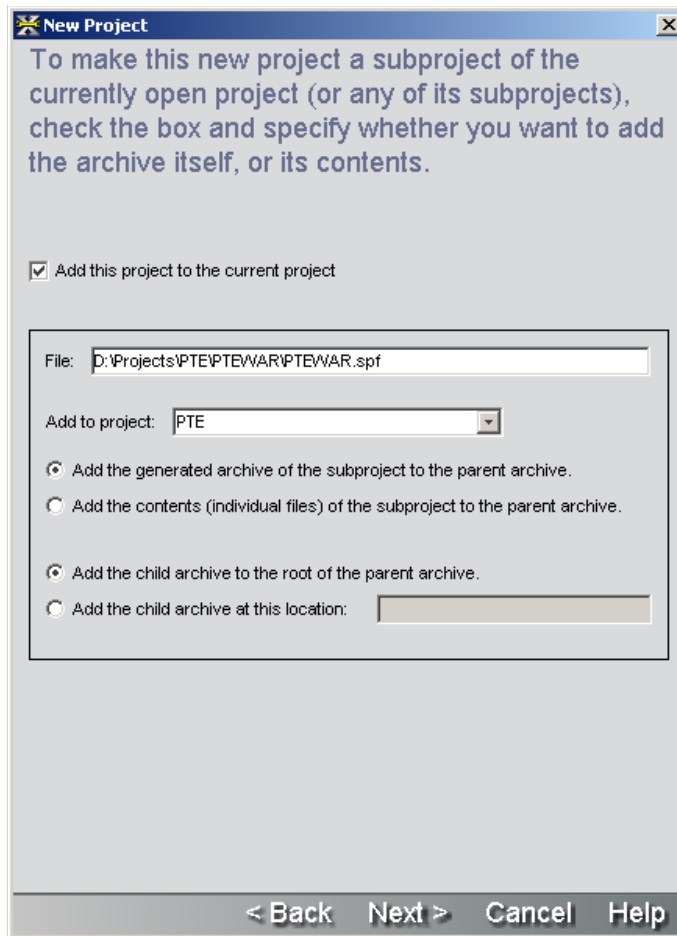
- **Create a Web Service SOAP trigger for the InventoryLookupWS Web Service**
 1. **File > New > Web Service > Composer Web Service > OK.** The Composer Wizard appears



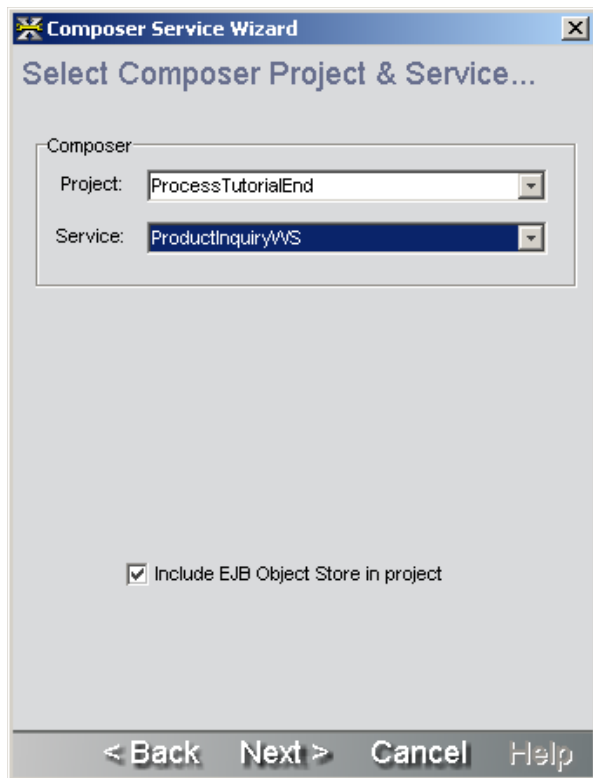
2. Press the Create WAR project button and the New Project dialog appears



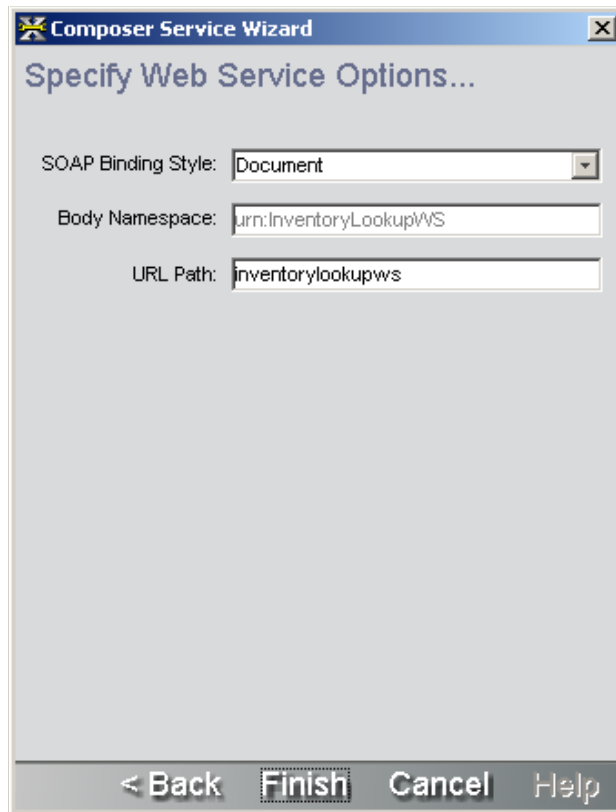
3. For Project Name enter: PTEWar
4. For Project Location enter the appropriate path
5. Make sure to select **j2ee 1.2** (ear1.2)
6. Answer **Yes** to create the new directory and the subproject panel appears



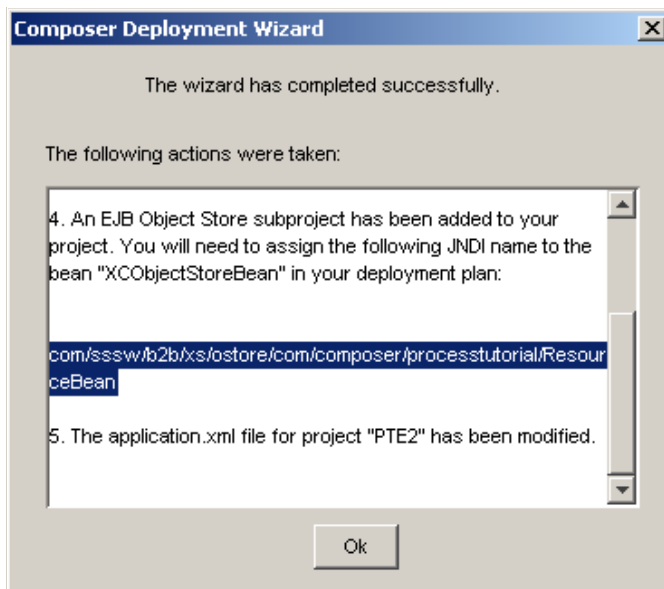
7. Press the **Next** button and then press the **Finish** button. The Composer Wizard returns
8. Press **Next** and the Select Project and Service dialog appears



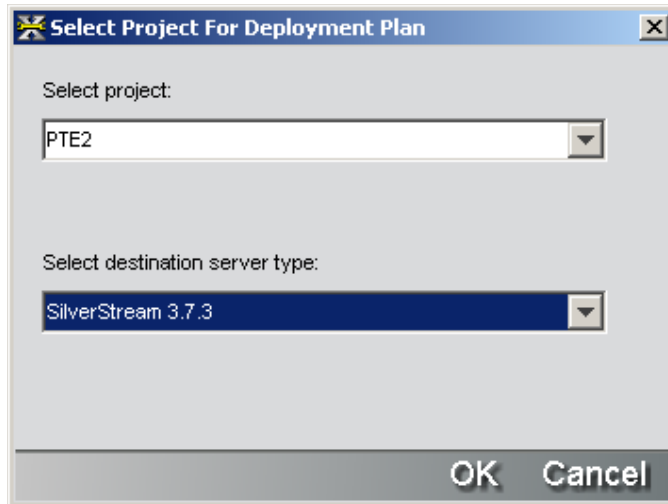
9. For Service select: InventoryLookupWS
10. Check the checkbox labelled “Include EJB Object Store in project”
11. Click **Next** and the Web Service Options panel appears



12. Click **Finish** and the Wizard Summary dialog appears



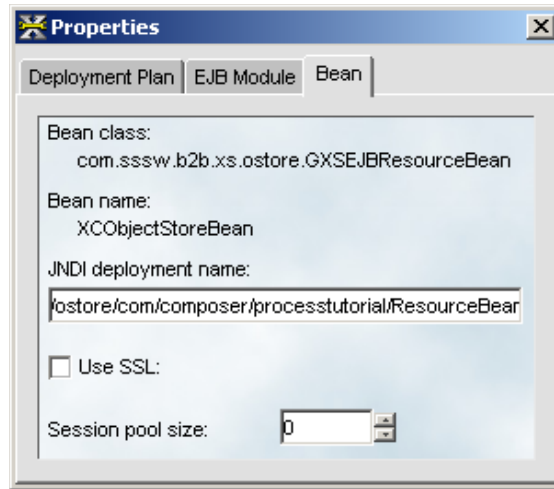
13. Scroll the message down and highlight the JNDI name
14. Press Ctrl-C to copy the string and press **OK** to dispatch the dialog
15. Create a Silverstream Deployment Plan
16. Select **File > New > Silverstream Deployment Plan** and press **OK** and the Select Project dialog appears



17. Set the destination server type to: Silverstream 3.7.3
18. Press **OK** and the Deployment Plan appears



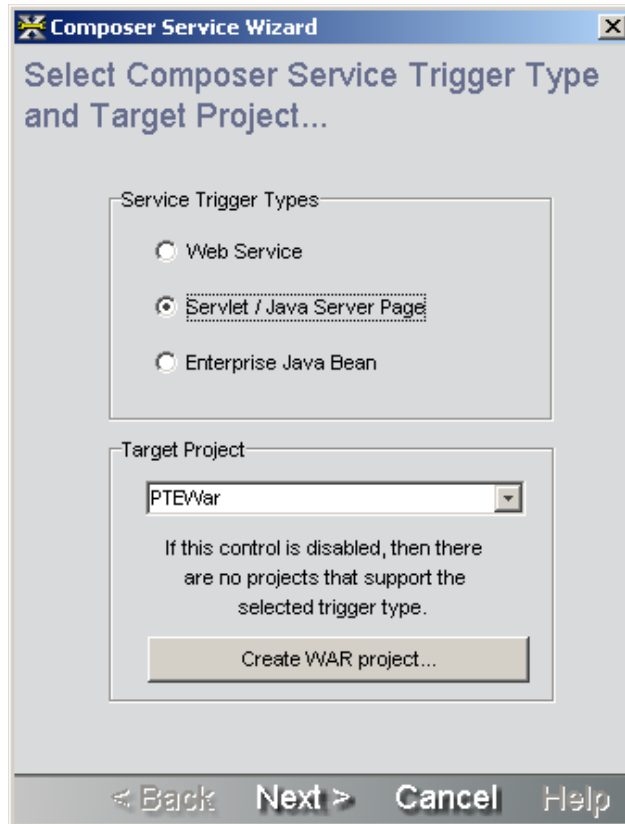
19. Select XObjectStoreBean and press the RMB (right mouse button)
20. Press **Properties** and the Properties dialog appears



21. In the JNDI deployment name field, press Ctrl-V to paste the JNDI name you copied
22. Close the Properties window
23. Close the Deployment plan and save it as PTEEARDeplPlan.xml

➤ **Create a Servlet trigger for the NotifyMe Service**

1. Select **File > New > Web Services > Composer Web Service** and press **OK**. The Trigger Type dialog appears



2. Select Servlet / Java Server Page
3. Press Next and the Select Service dialog appears
4. For the Service select: NotifyMe
5. Press Next and the Servlet Options panel appears
6. For the Name field enter: NotifyMe
7. For the URL Path enter: notifyme
8. Press Finish, OK and OK to complete the Servlet

➤ **Create a JSP trigger for the ProductInquiryWS Service**

1. Select **File > New > Web Services > Composer Web Service** and press **OK**. The Trigger Type dialog appears

2. Select Servlet / Java Server Page and press Next, and the Select Service panel appears
3. For the Service select: ProductInquiryWS and press Next, and the Servlet Options panel appears
4. For the Trigger select: JSP
5. For the Name enter: ProductInquiryWS
6. For the URL path enter: productinquiryws
7. For the Root Name enter: PRODUCTRESPONSE
8. Press Finish and the Done creating JSP dialog appears
9. Press OK to dispatch the dialog and OK again to dispatch the Wizard summary panel

➤ **Edit the JSP to create an Inquiry Form for the ProductInquiry service**

1. Press the RMB (right mouse button) in the text editor for the JSP and select Select All
2. Press the RMB again and select Cut
3. Copy the contents of the installed sample ProductInquiryWS.JSP into this JSP (the sample is in directory ..\Program Files\Silverstream\eXtend Composer\Samples\ProcessTutorialFiles)
4. Save the JSP

➤ **Create a Server Profile (if you don't already have one)**

1. Select **Edit > Profiles**, then the Servers tab
2. Press **New** and the Create a New Profile dialog appears
3. For Server Type select: Silverstream 3.7.3
4. Press **OK** to return to the Servers tab
5. Make sure the profile you just created appears in Profile Name

➤ **Deploy the project EAR**

1. Select **Project > Deployment Settings**
2. Select the profile you just created in Profile Name
3. Press **Deploy**.

➤ **Test the ProductInquiryWS service from the JSP page**

1. Open a Web Browser
2. In the address field type: <http://localhost:80/XCTutorial/PTEWar/ProductInquiryWS.jsp>