

Windows Workflow Foundation

Hands-On Lab

Lab Manual

Lab 06 – Workflows and Web Services in C#

Information in this document, including URL and other Internet Web site references, is subject to change without notice. This document supports a preliminary release of software that may be changed substantially prior to final commercial release, and is the proprietary information of Microsoft Corporation.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.

The entire risk of the use or the results from the use of this document remains with the user. Complying with all applicable copyright laws is the responsibility of the user.  Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document.  Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.

© 2005 Microsoft Corporation.  All rights reserved.

Microsoft, MS-DOS, Windows, Windows NT, Windows Server, Visual C# and Visual Studioare either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

**Contents**

[Overview 1](#_Toc218633763)

[Objectives 1](#_Toc218633764)

[System Requirements 1](#_Toc218633765)

[Setup 1](#_Toc218633766)

[Physical Folder Structure 1](#_Toc218633767)

[Code Snippets 2](#_Toc218633768)

[Starting Material 2](#_Toc218633769)

[Acronyms Used in this Lab 2](#_Toc218633770)

[Scenario 2](#_Toc218633771)

[Exercise 1 – Consuming Web Services 4](#_Toc218633772)

[Task 1 – Setup the Visual Studio solution 4](#_Toc218633773)

[Task 2 – Add the workflow project 6](#_Toc218633774)

[Task 3 – Add the workflow activities 7](#_Toc218633775)

[invokePOSubmissionWSFabrikam\_Invoking 16](#_Toc218633776)

[Task 4 – Configure your host to use the ContosoWorkflows 17](#_Toc218633777)

[Task 5 – Building and running the project 18](#_Toc218633778)

[Exercise 2 – Exposing Web Services 19](#_Toc218633779)

[Task 1 – Setting up the project 19](#_Toc218633780)

[Task 2 – Adding the Fabrikam workflow project 19](#_Toc218633781)

[Task 3 – Authoring the Fabrikam workflow project 20](#_Toc218633782)

[Task 4 – Publishing the workflow as a web service 22](#_Toc218633783)

[Task 5 – Accessing the published workflow 23](#_Toc218633784)

[Task 6 – Compiling and running the project 25](#_Toc218633785)

[Lab Summary 25](#_Toc218633786)

# Overview

Estimated time to complete this lab: **60 minutes**

## Objectives

The objective of this lab is to demonstrate how to consume web services from workflows and how to expose workflows as web services. There are two exercises:

* Exercise 1: Consuming a web service from a workflow

You will create a workflow to consume an existing web service in this exercise.

* Exercise 2: Exposing a workflow as a web service

You will create a workflow to be exposed as a web service and your new workflow will replace one of the web services used in the previous exercise.

More information about Windows Workflow Foundation can be found at <http://msdn.microsoft.com/workflow>

## System Requirements

* Microsoft Visual Studio 2008

## Setup

Unzip the lab to your local hard drive. Everything needed to complete the lab is in the zip file.

### Physical Folder Structure

File paths referenced in this lab assume the lab is installed in the following folder:

*C:\WF\WF 3.5 Labs\Lab[Number]*

Within the ***Lab[Number]*** folder, several child folders are available:

* ***CSharp*** – The lab written for C#
* ***VB*** – The lab written for VB
* ***resources*** –Any files referenced in the lab can be found in the Resources subdirectory, including source code for custom assemblies referenced in the exercises.

Within each *[Language]* folder, several child folders are available:

* ***before*** – The work area for completing the HOL
* ***after*** – The fully completed HOL

### Code Snippets

All code required for this lab consisting of more than 2 lines is available as code snippets. To learn more about code snippets including how to install them and how to use them, see the snippet guide document for the language of your choice in the folder:

*C:\WF\WF 3.5 Labs\Snippets*

## Starting Material

### Acronyms Used in this Lab

* WF – Windows Workflow Foundation

### Scenario

Contoso, which makes its procurements from Fabrikam, has set up an internal system to accept and process Purchase Order (PO) requests and running workflows for this task. Contoso has a very simple policy to process the PO’s, if the PO total is more than $1,000 the PO is routed to Northwind, otherwise, the PO gets sent to Fabrikam to be fulfilled.

PO has the following schema:

* PO Number
* PO Total
* Fulfiller correlation number (set by the fulfiller)
* PO history
  + - History item (a new history item is added each time PO is modified or processed)
      * Contact
      * Status
      * Time stamp
    - PO Items
      * PO Item
        + SKU
        + Amount
        + Price

Following is the basic business process flow for the PO processing systems for two likely outcomes:



PO is accepted and will be fulfilled by Fabrikam or Northwind.



PO is accepted, but fulfiller will not be able to fulfill it.

# Exercise 1 – Consuming Web Services

In this exercise, we will develop the workflow for the heart of Contoso’s system which processes the submitted PO’s and consumes a web service that exposes Fabrikam’s or Northwind’s PO processing service.

We will use a provided workflow host application and a web service that simulates the Fabrikam’s system.

Provided host application

PO processing workflow – Exercise code

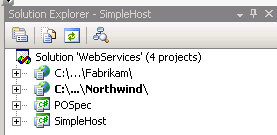
Provided Fabrikam web service

## Task 1 – Setup the Visual Studio solution

We will start with setting up the Visual Studio 2005 solution we will be using throughout the lab by copying the provided resources to the lab folder.

1. Open Windows Explorer and navigate to *C:\WF\WF 3.5 Labs\Lab06\Resources*
2. Copy all of the contents of this folder, to *C:\WF\WF 3.5 Labs\Lab06\*
3. Double click the Visual Studio solution file WebServices.sln to open Visual Studio 2005.
4. Visual Studio 2005 opens with the solution and the following should be present in the **Solution Explorer** window:

Fabrikam Web Service



Workflow host application

Helper project which contains the PO schema

Northwind Web Service

Important: The Web Services projects in the lab use the **ASP.NET Development Server** with the **Use Dynamic Ports** property set to *True* other than the default setting *False* and specific values are set for the **Port Number** properties of the projects. The environment this lab is run may be different on different computers, where this assumption may not be correct. If you observe inconsistent behavior with the web services please try changing the **Use Dynamic Ports** setting and/or the **Port Number** properties as a step of your troubleshooting process. The **Port Number** property values for the Web Service projects are set to the following values:

**Port Number** property value for Fabrikam Web Service: “*1582”*

**Port Number** property value for Northwind Web Service: “*2206”*

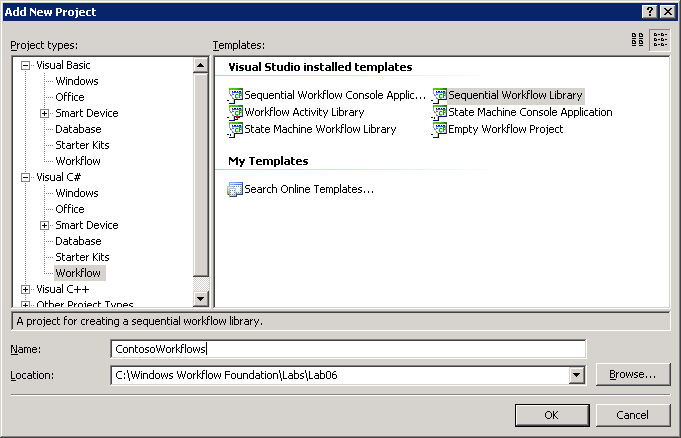
1. There are also completed versions of the exercises in the *Exercise 1* and *Exercise 2* folders under *Completed.* In order to run those exercises successfully, the **ASP.NET Development Server** instances have to be started. Please right-click on each of the Web Service projects in the solution and select **View in Browser** from the context menu to start the servers. Both exercises use two Web Services. If both of the servers are running, the flowing two icons should be in the **Notification Area** of the **Taskbar:**



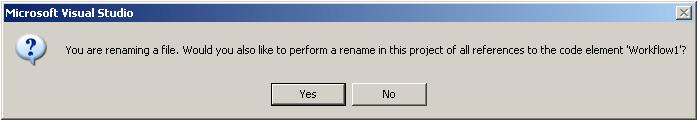
## Task 2 – Add the workflow project

The workflow project will create an assembly that contains the workflow model we will create. We will be using a sequential workflow.

1. In **Solution Explorer**, select the solution node *WebServices*, open the context menu by right-clicking, and select **Add | New Project.**
2. Visual Studio 2005 will display the **Add New Project** dialog window
3. In the **Add New Project** dialog window, expand **Visual C# | Workflow** in the Project Types tree on the left pane.
4. Select **Sequential Workflow Library** from the templates list and rename the project as ***ContosoWorkflows*** and click **OK**.



1. Add a project reference to the *POSpec* helper project in *ContosoWorkflows* by right-clicking on the project in the **Solution Explorer** view and selecting the **Add Reference** menu item.
2. Rename the **Workflow1.cs** workflow to **processPOWorkflow.cs** by right-clicking on it in the **Solution Explorer** and selecting **Rename***.* If the Visual Studio displays the following dialog, click *Yes*



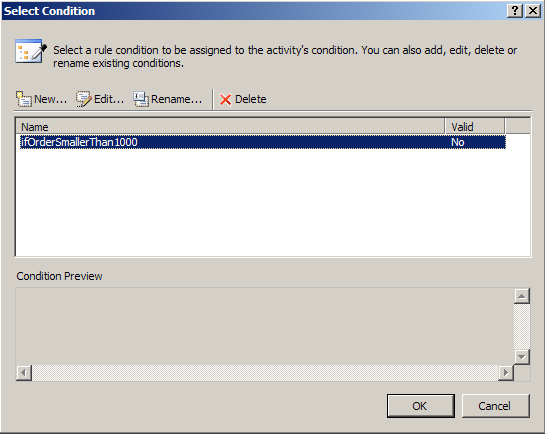
1. Move the file named *POHelpers.cs* under *C:\WF\WF 3.5 Labs\Lab06\Resources\* to*C:\WF\WF 3.5 Labs\Lab06\ ContosoWorkflows*
2. Right-click on the *ContosoWorkflows* project in the **Solution Explorer** and select **Add | Existing Item…** and add the file *POHelpers.cs* to the project.
3. Add the public member *newPO* to the workflow class by opening the *processPOWorkflow.cs* workflow. You can view the source code by right-clicking on the *processPOWorkflow.cs* design surface and selecting **View Code** menu item.

public POSchema.PO newPO = POSchema.PO.GeneratePOInstance();

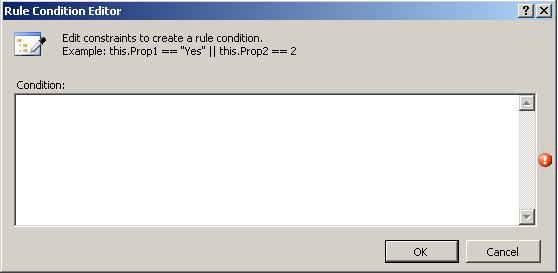
1. Compile the ***POSpec*** project by right clicking the project and selecting **Build.**

## Task 3 – Add the workflow activities

1. Switch to the designer view for the workflow *processPOWorkflow.cs***,** by double clicking it on the **Solution Explorer**.
2. Drag and drop an **IfElse** activity from the **Toolbox** to the Visual Studio workflow designer, just under the down-pointing green arrow.
3. Change the **Name** property of the **IfElse** activity to ***checkOrderTotal***.
4. Rename the *ifElseBranchActivity1* by selecting the decision branch and set the **Name** property to ***ifOrderSmallerThan1000***.
5. Select the condition type as **Declaritive Rule Condition** from the dropdown list for the **Condition** property.
6. Expand the **Condition** property node by clicking on the *+* sign and set the **ConditionName** property as ***ifOrderSmallerThan1000***.
7. Click on the Ellipsis button for the **ConditionName** property to display the **Select Condition** dialog box

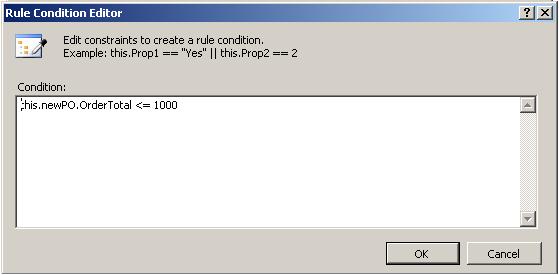


1. Select the **Edit Condition…** button to set the expression. This will open the **Rule Condition Editor**

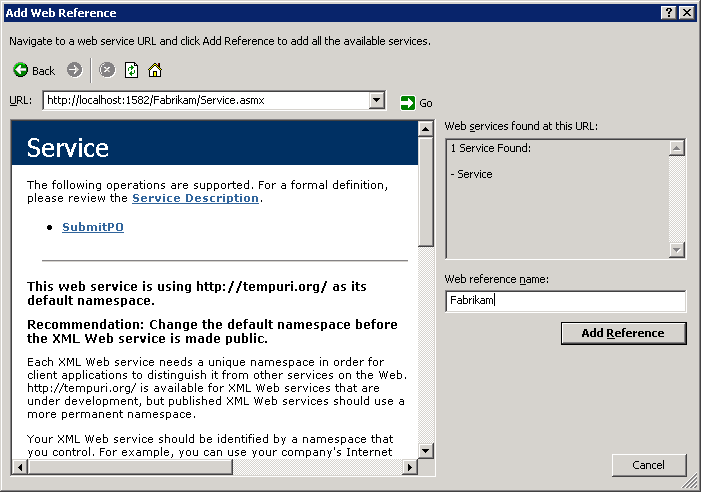


1. Add the following line into the **Rule Condition Editor**:

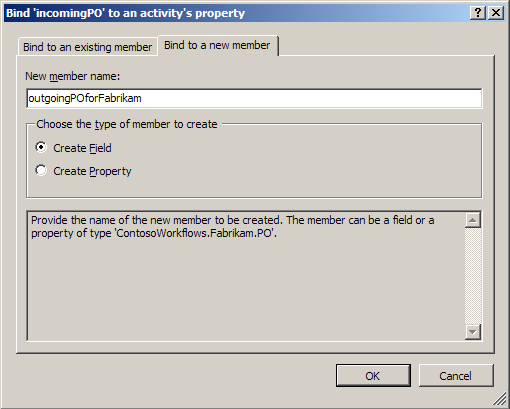
this.newPO.OrderTotal <= 1000



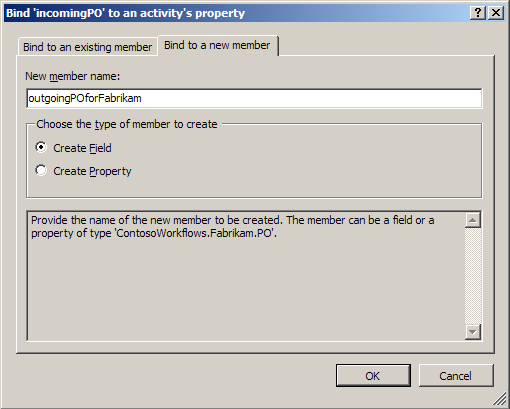
1. Click **OK** twice to close the **Select Rule Condition** and **Rule Condition Editor** dialog boxes.
2. Change the **Name** property of *ifElseBranch2* to ***Else***.
3. Right-click on the *Fabrikam* web service and select **Build Web Site.**
4. Right-click on the *Northwind* web service and select **Build Web Site**.
5. In the **Solution Explorer**, select the *Service.asmx* file in the *Fabrikam* web service project and right-click and select the **View In Browser** option in the context menu.
6. Copy the URL in the address bar by using **CTRL + C** in the opened Internet Explorer. Please do not forget to close the browser to be able to view the web service in the next step.
7. Make sure *processPOWorkflow.cs* is open by double clicking on it in Visual Studio 2005.
8. Drag and drop an **InvokeWebService** activity to the *ifOrderSmallerThan1000* branch. This will open the **Add Web Reference** dialog.
9. Paste the URL you have copied before into the URL field and press **Go** button.
10. Type in *Fabrikam* for the **Web Reference Name** text box.



1. Click on the **Add Reference** button to close the dialog box.
2. Change the Name property of the**InvokeWebService** activity you have added to *invokePOSubmissionWSFabrikam*.
3. Set the **Method Name** property of the *invokePOSubmissionWSFabrikam* activity by selecting ***SubmitPO*** from the dropdown list.
4. For the **incomingPO** property of the activity click the ellipsis button to bring up the activity binding interface, select the **Bind to a new member** tab and enter *outgoingPOforFabrikam* as the **New member name**.



1. Check the **Create Field** radio button and click **OK**.
2. For the **(ReturnValue)** property of the activity click the ellipsis button again to bring up the activity binding interface, select the **Bind to a new member** tab and enter *POResponsefromFabrikam* as the **New member name**.



1. Check the **Create Field** radio button and click **OK**.
2. These steps will add the following code to *processPOWorkflow.cs*:

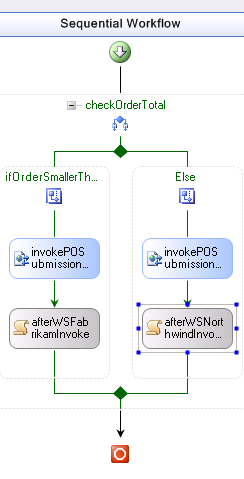
public ContosoWorkflows.Fabrikam.PO outgoingPOforFabrikam =

new ContosoWorkflows.Fabrikam.PO();

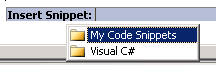
public ContosoWorkflows.Fabrikam.PO POResponsefromFabrikam =

new ContosoWorkflows.Fabrikam.PO();

1. Open the designer view, right-click on the activity *invokePOSubmissionWSFabrikam* and select **Generate Handlers….**to generate the Invoking code handler. We will copy the data to be passed to the web service in this step later with the code we will insert.
2. Switch to the design view again, drag and drop a **Code** activity from the **Toolbox** after the *invokePOSubmissionWSFabrikam* activity in the *ifOrderSmallerThan1000*branch and change its **Name** to ***afterWSFabrikamInvoke***.
3. In the **Solution Explorer**, select the *Service.asmx* file in the *Northwind* web service project and right-click and select the **View In Browser** option in the context menu.
4. Copy the URL in the address bar by using **CTRL + C** in the opened Internet Explorer. Please do not forget to close the browser to be able to view the web service in the next step.
5. Add another **InvokeWebService** for the *Else* branch
6. Repeat steps 15-26 for adding the invoke web service activity for *Northwind*. Use the following values for the activity’s properties
   1. Paste the URL as described in step 15.
   2. For step 19, use *Northwind* for the **Web Reference Name**
   3. For step 21, use *invokePOSubmissionWSNorthwind*as the activity’s **Name.**
   4. For step 22, use *SubmitPO* as the activity’s **Method Name**
   5. For step 23, use *outgoingPOforNorthwind*
   6. For step 25, use *POResponsefromNorthwind*
   7. Open the designer view, right-click on the activity *invokePOSubmissionWSNorthwind* and select **Generate Handlers….**to generate the Invoking code handler. We will copy the data to be passed to the web service in this step later with the code we will insert
7. Switch to design view again, add another **Code** activity from the **Toolbox** after the *invokePOSubmissionWSNorthwind*and change the **Name** property to *afterWSNorthwindInvoke*. Double click on the **Code** activities to generate their code handlers.
8. The resulting workflow should look like this:



1. If you are seeing red exclamation marks on the **Code** activities, that means the code handlers have not been not generated yet. Double-click on the **Code** activities to generate their code handlers.
2. Insert the code snippets specified in Table 1: Code handlers and snippets mappingby right-clicking in the code handlers and selecting **Insert Snippet…** navigate to the snippet you want and insert by double-clicking on **My Code Snippets** then open the *Lab06* snippets.



**Note:** If you do not see the Code Snippet in the list, then please make sure the code snippet files have been copied to the appropriate directory and have been added to the Code Snippets Manager (**Tools | Code Snippets Manager**)

Table : Code handlers and snippets mapping

|  |  |
| --- | --- |
| Code handler | Snippet Name |
| invokePOSubmissionWSNorthwind\_Invoking | WFLab06\_Ex01\_Task03\_invokePOWSNorthwind |
| invokePOSubmissionWSFabrikam\_Invoking | WFLab06\_Ex01\_Task03\_invokePOWSFabrikam |
| afterWSFabrikamInvoke\_ExecuteCode | WFLab06\_Ex01\_Task03\_afterWSFabrikamEx |
| afterWSNorthwindInvoke\_ExecuteCode | WFLab06\_Ex01\_Task03\_afterWSNorthwindEx |

If you have created the member variables with different names during the previous steps, please make sure to modify the code snippets accordingly.

**invokePOSubmissionWSNorthwind\_Invoking**

outgoingPOforNorthwind = new Northwind.PO();

POHelpers.CopyPOHeader(newPO, outgoingPOforNorthwind);

POHelpers.CopyPOHistory(newPO, outgoingPOforNorthwind);

POHelpers.CopyPOItems(newPO, outgoingPOforNorthwind);

### invokePOSubmissionWSFabrikam\_Invoking

POHelpers.CopyPOHeader(newPO, outgoingPOforFabrikam);

POHelpers.CopyPOHistory(newPO, outgoingPOforFabrikam);

POHelpers.CopyPOItems(newPO, outgoingPOforFabrikam);

**afterWSNorthwindInvoke\_ExecuteCode**

// Initialize it to a time where electronic PO processing was not invented.

DateTime statusDate = new DateTime(1900, 1, 1);

Northwind.POStatus lastStatus = null;

//Find the last status by looping over the history items

foreach (Northwind.POStatus stat in POResponsefromNorthwind.History)

{

if (stat.Timestamp > statusDate)

{

statusDate = stat.Timestamp;

lastStatus = stat;

}

}

System.Console.WriteLine("Your PO has been {0} on {1} by Northwind", lastStatus.PoStatus, lastStatus.Timestamp);

**afterWSFabrikamInvoke\_ExecuteCode**

// Initialize it to a time where electronic PO processing was not invented.

DateTime statusDate = new DateTime(1900, 1, 1);

Fabrikam.POStatus lastStatus = null;

//Find the last status by looping over the history items

foreach (Fabrikam.POStatus stat in POResponsefromFabrikam.History)

{

if (stat.Timestamp > statusDate)

{

statusDate = stat.Timestamp;

lastStatus = stat;

}

}

System.Console.WriteLine("Your PO has been {0} on {1} by Fabrikam", lastStatus.PoStatus, lastStatus.Timestamp);

**Note:** The code snippets you have just inserted convert between two different data types of PO for the internal Contoso system and the PO type exposed by the web service. The schema for the two PO types is the same though**.**

## Task 4 – Configure your host to use the ContosoWorkflows

1. Right-click on the *SimpleHost* project in the **Solution Explorer** and add a project reference to *ContosoWorkflows* project (your newly created Workflow project)
2. Open the file *Program.cs* by double-clicking on it in the solution explorer.
3. **Build** *SimpleHost* by right clicking *Simplehost* and choosing **Build**. The build will fail at this time but will allow us to continue with step 4.
4. Make sure that you can view the **Tasks List** window in Visual Studio 2005 by selecting **View | Task List** (or by **CTRL+\, CTRL+T**)
5. Select **Comments** from the drop down box in the **Task List** (default view is **User Tasks**) to view the places you will be making modifications in the *SimpleHost* project code.
6. Double click on the task that reads *TODO: Add a line to the code similar to the one below*
7. Add the code line as shown to specify the workflow type you will use.

Type type = typeof(ContosoWorkflows.processPOWorkflow);

## Task 5 – Building and running the project

1. Right-click and select **Build Solution** on the *WebServices* solution node in the **Solution Explorer** or press **CTRL+SHIFT+B** to rebuild the whole solution.
2. Right-click on the *SimpleHost* project and select **Set as StartUp Project**

Note: If the solution does not run, please check whether the port numbers for the web services are the same as the ones in the web references for the invoke web service activities in the workflow.

1. Press **CTRL+F5** to run the SimpleHost. The output should be like the following in a console window.

|  |
| --- |
| Your PO has been Accepted on 6/18/2005 6:21:25 PM by Fabrikam  Press Enter to continue... |

1. Press **ENTER** twice to close the console window.

# Exercise 2 – Exposing Web Services

We will build on our previous work for this exercise, replacing the *Fabrikam* web service with a new workflow.

Provided host application

PO processing workflow – from Exercise 1

Fabrikam web service will be replaced

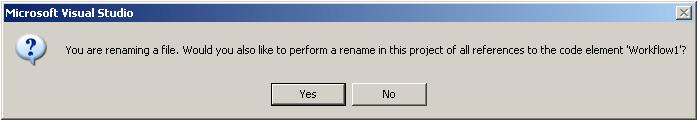
Exercise 2 structure

## Task 1 – Setting up the project

1. If the solution from the previous exercise is not open, open it now.
2. Right-click on the *Fabrikam* web service in the **Solution Explorer** and select **Remove**.

## Task 2 – Adding the Fabrikam workflow project

1. In **Solution Explorer**, select the *WebServices* solution node, and open the context menu by right-clicking, and select **Add | New Project.**
2. Visual Studio 2005 will display the **Add New Project** dialog window
3. In the **Add New Project** dialog window, expand **Visual C# | Workflow** in the **Project Types** tree on the left pane.
4. Select **Sequential Workflow Library** from the templates list and rename the project as ***FabrikamWorkflows*.**
5. Add a reference to project*POSpec*in the newly created project.
6. Rename the *Workflow1.cs* workflow to *processPOWorkflow.cs* by right-clicking on it in the **Solution Explorer** and selecting **Rename**.
7. If the Visual Studio displays the following dialog, click *Yes*



## Task 3 – Authoring the Fabrikam workflow project

1. Open *processPOWorkflow.cs* workflow by double clicking it.
2. Right-click on the design surface and select **View Code**
3. Add the following interface definition to the namespace:

(Snippet: “WFLab06\_Ex02\_InterfaceDefinition”)

//Webservice interface definition for the workflow

public interface IProcessReceivedPO

{

POSchema.PO ReceiveAndProcessPO(POSchema.PO aPO);

}

1. This interface defines the interface for the workflow to be exposed as a web service.
2. Switch to the design view by double-clicking on the *processPOWorkflow.cs*. Drag a **WebServiceInput** activity from the **Toolbox** and drop it just under the down-pointing green arrow on the design surface. Change its **Name** property to ***webServiceReceivePO***.
3. Drag and drop a **WebServiceOutput** activity from the **Toolbox** below the **webServiceReceivePO** activity and change the **Name** property to ***webServiceResponsePO***.
4. Set the **Interface Type** property of the *webServiceReceivePO* activity to *FabrikamWorkflows.IProcessReceivedPO* by clicking on the Ellipsis button in the property value box and selecting the interface from the type picker dialog box.
5. Select the *webServiceReceivePO* activity and set the activity properties as follows:
   1. **IsActivating**: *True*
   2. **MethodName**: *ReceiveAndProcessPO*
6. For the **aPO** property, click the ellipsis to open the activity binding interface and bind aPO to a new field member named *receivedPO****.***
7. Select the *webServiceResponsePO* activity and set the **InputActivityName** to *webServiceReceivePO.*
8. For the **(ReturnValue)** property, click the ellipsis to open the activity binding interface and bind **(ReturnValue)** to a new field member named *returnedPO****.***
9. Right click on the *webServiceResponsePO* and select **Generate Handlers…** to add the *SendingOutput*code handler.
10. Add the following code to the *webServiceResponsePO\_SendingOutput* code handler.

(Snippet: “WFLab06\_Exercise02\_Task03\_webServiceResponsePO”)

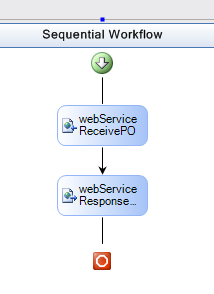
POSchema.PO.GenerateResponseHeader(receivedPO, returnedPO, "Fabrikam\_");

POSchema.PO.CopyHistoryAndChangeStatus(receivedPO, returnedPO, "Accepted");

POSchema.PO.CopyPOItems(receivedPO, returnedPO);

Note: It is extremely important to have defined responsePO and receviedPO exactly as they were described in the previous steps, as this code assumes those members exist.

1. The resulting workflow should look like this:



## Task 4 – Publishing the workflow as a web service

1. Build the solution to make sure that it compiles successfully
2. Right-click on the *FabrikamWorkflows* project in the **Solution Explorer** and select **Publish as web service**
3. Observe that this action created a new project in the same solution with the name *FabrikamWorkflows\_WebService*
4. This action will cause the following to occur:
   1. Creation of an .asmx file that specifies a web service deriving from *FabrikamWorkflows.processPOWorkflow\_WebService*

<%@WebService Class="FabrikamWorkflows.processPOWorkflow\_WebService" %>

* 1. Modification of the HTTPPipeline for the service for hosting workflows and running them

<httpModules>

<add type="System.Workflow.Runtime.Hosting.WorkflowWebHostingModule, System.Workflow.Runtime, Version=3.0.00000.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" name="WorkflowHost"/>

</httpModules>

* 1. Out-of-box pluggable runtime services configuration in the web.config file

<configSections>

<section name="WorkflowRuntime" type="System.Workflow.Runtime.Configuration.WorkflowRuntimeSection, System.Workflow.Runtime, Version=3.0.00000.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />

</configSections>

<WorkflowRuntime Name="WorkflowServiceContainer" >

<Services>

<add type="System.Workflow.Runtime.ManualWorkflowSchedulerService, System.Workflow.Runtime, Version=3.0.00000.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35” />

<add type="System.Workflow.Runtime.Hosting.DefaultWorkflowTransactionService, System.Workflow.Runtime, Version=3.0.00000.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />

</Services>

</WorkflowRuntime>

1. This Lab uses fixed port numbers for the Web Services. Please set the **Use Dynamic Ports** property of the newly generated Web Service project to *False* and set the **Port Number** to “*4487”*.

Note: If you observe inconsistent behavior with the Web Services, changing the “Use Dynamic Ports” property value to “True” can be one of the troubleshooting steps, as the port assignment assumptions of this lab may be conflicting with the environment.

## Task 5 – Accessing the published workflow

1. If a browser displaying a web service from a previous step is still open, close the browser first. Right-click on the newly generated web service (.asmx file) and select **View in Browser**. Click on the *.asmx* resource in the browser and copy the **URL** string in the **Address Bar** of the browser using **CTRL+C**.
2. Open the *processPOWorkflow.cs* of the *ContosoWorkflows* project by double-clicking it in the **Solution Explorer**.
3. Delete the Web Reference name *Fabrikam* from the *ContosoWorkflows* project, by extending the **Web References** node of the project, selecting the Web reference *Fabrikam*, right clicking on it and selecting **Delete** from the context menu.
4. Delete *invokePOSubmissionWSFabrikam* by selecting it and right clicking then selecting **Delete**.
5. Drag an **InvokeWebReference** from the **Toolbox** to the position *invokePOSubmissionWSFabrikam* **previously held.**
6. Paste the **URL** string you have copied in step 1 into the **URL** text box in the **Web Reference** box and press **Go** button. Set the **Web Reference Name** to *Fabrikam***.** Press the **Add Reference** button to update the web reference.
7. Change the **Name** property to *invokePOSubmissionWSFabrikam*
8. Change the following properties with the provided values
   1. **ProxyClass**: ContosoWorkflows*.Fabrikam.processPOWorkflow\_WebService*
   2. **Method:** *ReceiveAndProcessPO*
9. For the **(ReturnValue)** property, click the ellipsis to open the activity binding interface and bind **(ReturnValue)** to the existing field member named *POResponsefromFabrikam****.***
10. For the **aPO** property, click the ellipsis to open the activity binding interface and bind **aPO** to the existing field member named *outgoingPOforFabrikam****.***
11. In the **Properties** panel click the **Events** button. This will display the **Invoked** and **Invoking** handlers. Set the following:

**Invoked**: *invokePOSubmissionWSFabrikam\_Invoked*

**Invoking**: *invokePOSubmissionWSFabrikam\_Invoking*

1. **Delete** the **Web Reference Name** ***Northwind*** from the *ContosoWorkflows* project, by extending the **Web References** node of the project, selecting the Web reference *Northwind*, right clicking on it and selecting **Delete** from the context menu.
2. In the **Solution Explorer**, select the *Service.asmx* file in the *Northwind* web service project and right-click and select the **View In Browser** option in the context menu.
3. Copy the **URL** in the **Address Bar** by using **CTRL + C** in the opened Internet Explorer. Please do not forget to close the browser to be able to view the web service in the next step.
4. Right click the *ContosoWorkflows* project and select **Add WebReference**.
5. Paste the **URL** string you have copied in step 14 into the **URL** text box in the **Web Reference** box and press **Go** button. Set the **Web Reference Name** to *Northwind***.** Press the **Add Reference** button to update the web reference.  
     
   NOTE: The following additional steps may be necessary because removing the Northwind web service reference sometimes clears these settings:
6. Select *invokePOSubmissionWSNorthwind*
7. Set the following properties:
   1. **ProxyClass**: *ContosoWorkflows.Northwind.Service*
   2. **MethodName**: *SubmitPO*
   3. **(ReturnValue)**: *POResponsefromNorthwind*
   4. **incomingPO**: *outgoingPOforNorthwind*

## Task 6 – Compiling and running the project

1. **Build** the solution as described in of Exercise 1 at step 1.
2. Press **CTRL+F5** to run the solution (please make sure that *SimpleHost* project is set up as StartUp project)
3. You should be able to see the same result as in Exercise 1.
4. Press **ENTER** twice to close the console window

# Lab Summary

The objective of this lab was to introduce consuming web services from the workflows and exposing them as web services. Using Web Services is one of the mechanisms to communicate with the external applications from within the workflows. We completed two exercises;

The first one was consuming two web services. Contoso had a simple policy, if the PO total is larger than $1000 it would fulfill the PO through Northwind, otherwise, it is always Fabrikam. The PO gets submitted to the corresponding web service from within the workflow.

We changed the implementation of Fabrikam’s web service during the second exercise with a workflow. We then expose Fabrikam’s workflow which implements the business logic as a web service to be consumed from within Contoso’s workflow.