Hands-On Labs

Introduction to Workflow Services using .NET Framework 4

Lab version: 1.0.0
Last updated: 12/10/2010
CONTENTS

OVERVIEW ........................................................................................................................................... 4

SETUP .................................................................................................................................................. 6
  Task 1 – Installing the Web Platform, Visual Studio and Windows Server AppFabric .................. 7
  Task 2 – Creating the Databases ........................................................................................................ 8
  Task 3 – Installing and Configuring Windows Server AppFabric .................................................. 8

EXERCISE 1: SUBMIT APPLICATION SERVICE ............................................................................. 11
  Task 0 – Opening the Solution ........................................................................................................... 11
  Task 1 – Adding a new a Workflow Service ....................................................................................... 11
  Task 2 – Exploring the Workflow Service .......................................................................................... 13
  Task 3 – Creating Service Implementation Outline .......................................................................... 15
  Task 4 – Adding Custom Activities .................................................................................................. 18
  Task 5 – Receiving a Job Application ............................................................................................... 19
  Task 6 – Saving the Job Application .................................................................................................. 22
  Task 7 – Configuring the Response ................................................................................................... 23
  Task 8 – Adding the Client Application ............................................................................................. 24
  Exercise 1: Verification ....................................................................................................................... 26

EXERCISE 2: EMAIL NOTIFICATION ............................................................................................ 28
  Task 0 – Opening the Solution ........................................................................................................... 28
  Task 1 – Modifying the Workflow Service .......................................................................................... 28
  Exercise 2: Verification ....................................................................................................................... 29

EXERCISE 3: APPLICANT SCREENING .......................................................................................... 31
  Task 0 – Opening the Solution ........................................................................................................... 31
  Task 1 – Adding the Education Screening Activity ............................................................................ 31
  Task 2 - Calling the ScreenEducation Activity ................................................................................... 32
  Task 3 – Conditionally Notifying a Human Reviewer ....................................................................... 33
  Task 4 – Sending Human Screening Results .................................................................................... 36
  Exercise 3: Verification ....................................................................................................................... 38

EXERCISE 4: DEPLOY THE SOLUTION ........................................................................................... 40
  Task 0 – Opening the Solution ........................................................................................................... 41
  Task 1 – Creating Configuration Transformations ............................................................................ 41
Overview

Windows Workflow Foundation (WF4) in .NET 4 introduces a new way to create web services that combines the power of Windows Communication Foundation (WCF), and the flexibility of WF4. In this lab, you will learn how to use the WF4 messaging activities to implement an HR business workflow scenario. You will also experience the improved hosting, deployment, tracking and troubleshooting capabilities of the new Windows Server AppFabric.

Do I have to do all the exercises?

This lab is long. To do all of it will probably take close to 2 hours. The lab is designed so that each exercise can be done independently. Just open the solution found under the Begin folder for the exercise. For example, if you wanted to learn how Windows Server AppFabric make it easier to deploy an application you would start with Exercise 5 and open the solution from the \Source\Ex5\Begin (choosing the folder that matches the language of your preference) folder.

Contoso has a business process for filling the open job requisitions in the company. The process will be divided in three tiers: a smart client application, a web site, a middle-tier workflow service, and a data layer.

1. The applicant submits an application to a workflow service using a Windows Forms client app
2. The workflow service screens the applicant’s education background by calling an Education service
3. If the education screening passes, a human reviewer will receive an email with a link to the website to review the application
4. The web site will call the workflow service to pass the result of the human review
5. The applicant is notified by an email and a callback to the client application
Figure 1
*Services Diagram for the lab solution*

**Objectives**

In this Hands-On Lab, you will learn how to:

- Use Messaging activities to send data in and out of workflows
- Configure the Messaging activities for a specific scenario
- Invoke Add Service Reference to create activities that consume another service
- Utilize Activity Templates to create pre-configured request-reply patterns
- Set up Content Based Correlation to correlate messages from the same client

**System Requirements**

You must have the following items to complete this lab:

- Microsoft Visual Studio 2010 - [Download](#)
- Microsoft .NET Framework 4 - [Download](#)
- SQL Server 2008 Express (included with Visual Studio 2010)
- Microsoft Outlook or Windows Live Mail Client - [Download](#)
- Windows Server AppFabric (Beta 2) – Download using the Web Platform Installer (see below)
Note: This Lab targets Visual Studio 2010 and .NET Framework 4. The current version of Windows Server AppFabric, that is Beta 2, requires .NET Framework 4 RC. Therefore, the portion of this Lab that shows AppFabric features may not work as expectedly until a new version of AppFabric is released.

Exercises

This Hands-On Lab comprises the following exercises:

1. Submit Application Service
2. Email Notification
3. Applicant Screening
4. Deploy the Solution
5. Troubleshooting

Starting Materials

This Hands-On Lab includes the following starting materials.

- Visual Studio solutions. Depending on the exercise you will find Visual Studio solutions that you can use as starting point for the exercises.

What if I get stuck?

The source code that accompanies this hands-on lab includes an end folder where you can find a Visual Studio solution with the code that you would obtain if you complete the steps in each exercise. You can use this solution as a guide if you need additional help working through the exercises.

Remember to build the solutions before opening any file using the workflow designer.

Next Step

Setup

Setup
Task 1 – Installing the Web Platform, Visual Studio and Windows Server AppFabric

1. Download the Microsoft Web Platform installer

2. Install the recommended Web Platform options plus the following
   a. ASP.NET
   b. .NET Extensibility
   c. Web Deployment Tool 1.0
   d. Windows Server AppFabric Beta 2

   **Note:** In the Web Platform Installer Windows Server AppFabric shows up in the Enterprise tab. This tab is not displayed by default. To display it, select options and tick the box for Enterprise scenarios. Afterward you will see the option for Application Server in the Enterprise tab.

3. Download and Install Visual Studio 2010 and .NET Framework 4
4. You can optionally download and Install Windows Server AppFabric without using the Web Platform Installer.
   a. Configure both the Persistence and Monitoring database
   b. Accept all the defaults on the remaining install pages

Task 2 – Creating the Databases
1. Run the Dependency checker for the Training Kit if you haven't done it previously. To do this, browse to the Source\Setup folder of this lab, and run the CheckDependencies.cmd script. Install any pre-requisites that are missing (rescanning if necessary) and complete the wizard.

   Note: This script file creates a persistence database by invoking the SQL scripts included with the .NET Framework. You can also use AppFabric to create persistence databases.

After installing Windows Server AppFabric, you will be prompted to configure it. If you want to reconfigure it later just search for the Configure AppFabric menu shortcut. AppFabric can be configured to work with The SQLExpress database created by Visual Studio so you do not need to install a separate SQL Server.

   Note: The Web Platform Installer automatically configures Windows Server AppFabric so you do not need to do these steps again unless you want to re-configure AppFabric.

1. Start Configure AppFabric from the Start menu if it is not already running after installation.
2. On the Configure Hosting Services page tick the Set monitoring configuration checkbox
4. Tick the Set persistence configuration checkbox
5. Select the persistence provider sqlStoreProvider.
6. Right click the **DefaultMonitoringConnectionString**, and choose **Initialize Database** ... Use the defaults and click the **OK** button.
Register and Initialize the monitoring store

7. Click the back arrow to return to the main screen.

8. In the Features View pane in the center double click the Persistence Database Configuration icon.

9. Right-click the DefaultPersistenceConnectionString, and choose Initialize Database. Use the defaults, and click the OK button.

Next Step

Exercise 1: Submit Application Service
Exercise 1: Submit Application Service

In the first exercise you will create a simple service that receives an Application from the smart client application and returns a response.

Task 0 – Opening the Solution


2. Open the starting solution for Exercise 1 located under the Source\Ex1\Begin folder (choosing the folder that matches the language of your preference.) Use it as the starting point for this exercise.

   What is in this solution?

   This solution contains a web site created from the WCF Workflow Service template, and a data layer project you will be using in this lab. The web.config file has some minor modifications to include an appSettings section, and connection string for the database.

3. Press CTRL+SHIFT+B to build the solution

Task 1 – Adding a new a Workflow Service

1. Add a new Workflow Service. To do this, right click on HRApplicationServices, and select Add / New Item... (Ctrl+Shift+A), in the Workflow Templates select the WCF Workflow Service template, name the service to SubmitApplication.xaml, and then click Add.
Figure 5
Create a New Workflow Service in C#

Figure 6
Create a New Workflow Service in Visual Basic
Task 2 – Exploring the Workflow Service

You probably have some questions about your service. What is the service contract? What binding does it use? What address is it hosted at? In this task, you will explore your service and how it interacts with the world.

Figure 7
The default Workflow Service

1. Workflow Services have an implicit contract that is defined by the messaging activities contained in the service. SubmitApplication.xaml already has a working contract that might look familiar if you have built a WCF service before. Look at SubmitApplication.xaml in the workflow designer. Notice that it has a sequence with two activities, a request labeled ReceiveRequest and a response labeled SendResponse. The request determines the parameters to the service. To see the message content, click View message in the ReceiveRequest activity, next to the Content label.
2. Click **OK** to dismiss the Content Definition dialog.

3. The content of the message is a variable named data. To see this variable; click the Sequential Service activity and then click the **Variables** button in the lower left of the design surface.

4. There are two variables. The first variable **handle** is used to correlate the request and response messages. The second variable **data** defines the message content that you expect to receive.

5. To see what the return value is, in the **SendResponse** activity click **View message**.
6. Click OK to dismiss the Content Definition dialog.

7. You now have a workflow service, but it does not have Metadata publishing enabled. To add this, you can use the new “tagless” syntax for WCF configuration in .NET 4. Open the web.config file, and add the following section. Also notice that there is no <service> definition. The simplified configuration for WCF will supply a default binding (basicHttpBinding) address and contract as required.

(Code Snippet - Introduction to WFServices Lab – WCF serviceMetadata XML)

```xml
<system.serviceModel>
  <behaviors>
    <serviceBehaviors>
      <behavior>
        <serviceMetadata httpGetEnabled="true"/>
      </behavior>
    </serviceBehaviors>
  </behaviors>
</system.serviceModel>
```

8. You now have a working Workflow Service. To see it in action, right click the SubmitApplication.xaml file, and select View in Browser (Ctrl+Shift+W). You should see a familiar site for WCF developers.

Figures 10 and 11

The response message will be the data converted to a string

---

Task 3 – Creating Service Implementation Outline
Your service has three phases to it. It will receive, screen and notify. Throughout the lab you will be implementing these phases. To begin with, you will add placeholders for each phase.

1. Switch back to `SubmitApplication.xaml`.

2. Change the Display Name property of the outermost sequence from *Sequential Service* to *Application Service*. You can change it directly on the design surface or in the properties window.

![Application Service](image)

**Figure 12**

*Change the name of the service to Application Service*

3. The first placeholder will require a transaction because you will be updating a database as well as persisting your workflow to the persistence database. Drag a *TransactedReceiveScope* from the *Messaging* group in the toolbox and drop it at the top of the Application Service sequence. Set the Display Name to *Receive Application*.

4. Drag the existing *ReceiveRequest* inside the Request area of the Receive Application activity. When the message is received it will create a transaction.

![Application Service](image)
Figure 13
*Drag the existing ReceiveRequest inside of the Request area of the Receive Application*

5. In the body area you want to put the work that will be done inside of the transaction. You will have more than one activity here so drop a **Sequence** from the **Control Flow** toolbox group inside of the body and set the properties
   a. Display Name: **Save and Respond**

6. Drag the **SendResponse** and drop it inside of the Save and Respond sequence.

Figure 14
*The Service with the TransactedReceiveScope containing the messaging activities*

8. The second phase of the service will be to screen the applicant. You will implement this later but for now drag a **Sequence** activity from the **Control Flow** group and drop it below the Receive Application shape and set the Display Name to **Screen Applicant** then collapse the sequence.

9. The third phase will be notifying the applicant via email and by sending a notification message to the smart client application. Drop another Sequence, set the Display Name to **Notify Applicant** and collapse the sequence.

![Collapsible view of workflow service](image)

**Figure 15**
*Collapsed view of the workflow service with three phases*

---

**Task 4 – Adding Custom Activities**

Your service will accept a job application from the smart client app. In this task you will make the modifications necessary to prepare the service to accept a job application.

1. Add existing projects containing the solution's data contracts and custom activities. To do this, in Solution Explorer right-click the solution node, and select **Add / Existing Project**. In the Open Project dialog, browse to **Source\Assets** folder of this lab (choosing the folder that matches the language of your preference) and select the following project(s)

   a. **HRApplicationServices.Contracts**
b. HRApplicationServices.Activities

2. In the HRApplicationServices project, add references to the HRApplicationServices.Contracts and HRApplicationService.Activities projects. To do this in Solution Explorer, right-click the HRApplicationServices project and select Add Reference. In the Add Reference dialog, switch to the Projects tab, and select the projects. Click OK to add the references.

3. Your workflow is going to be long running. It will do some initial work and then wait for a human to respond. This means that you need to configure a persistence store. To add this, open the web.config from the HRApplicationServices project and add the following settings.

   **Note:** When you deploy the website to AppFabric you can configure the persistence store within IIS manager. While working with the ASP.NET Development server you have to create the persistence stores and add the persistence behavior manually.

   a. To the <connectionStrings> section add the following connection string which will use the WF4Persistence store you created in the setup for the lab.

   (Code Snippet - Introduction to WFServices Lab – WF4 Persistence XML)

   ```xml
   <connectionStrings>
     <add name="WF4Persistence"
         connectionString="Data Source=\sqlexpress;
         Database=WF4Persistence;
         Integrated Security=True"/>
   </connectionStrings>
   ```

   b. To the <behavior> section add the sqlWorkflowInstanceStore behavior right after the serviceDebug behavior.

   (Code Snippet - Introduction to WFServices Lab – WF4 sqlWorkflowInstanceStore XML)

   ```xml
   <serviceMetadata httpGetEnabled="true"/>
   <sqlWorkflowInstanceStore connectionStringName="WF4Persistence" />
   ```

4. To make the custom activities available to your service you must build the solution. From the menu select Build / Build Solution (Ctrl+Shift+B). Ensure that the build succeeded by inspecting the Output window. If you look at SubmitApplication.xamlx in the workflow designer, you should now see a HRApplicationService.Activities group in the toolbox.

---

**Task 5 – Receiving a Job Application**
1. Open `SubmitApplication.xaml` and create variables to store the application request and response. To do this, follow these steps:

   a. Select the **Application Service** activity by clicking over the shape surface.

   b. Click the **Variables** button located in lower left side of the workflow designer. A panel that displays the available variables for the **Application Service** activity appears.

   c. Delete the `data` variable by selecting it and pressing the **Delete** key.

   #### Note:
   When you delete the `data` variable the workflow designer will show warning icons indicating that the workflow is invalid until you update the activities that used the `data` variable.

   d. Click **Create Variable**.

   ![Create Variable](image)

   **Figure 16**
   *Adding a new variable*

   e. Type **ApplicationRequest** in the Name box.

   f. In the Variable type drop-down list select **Browse for Types**.

   g. In the **Type Name** box of the dialog box, type the first few characters of the type you are looking for such as **SubmitJob**. As you type matching names will show up in the list. Click on the **SubmitJobApplicationRequest** type to select it.
h. Click OK to finish.

2. Repeat the previous process to add another variable named ApplicationResponse of type SubmitJobApplicationResponse when finished click the Variables button to hide the variables window

3. You need to create a response object to store the response. Set the Default value of the ApplicationResponse variable to New SubmitJobApplicationResponse(). Remember that expressions must be written with Visual Basic regardless of the language of the project.

4. Now you can change the contract to receive a job application. To do this, expand the Receive Application sequence and select the ReceiveRequest activity in the designer and press F4 to display the Properties Window. Set the following values:
   a. OperationName: SubmitJobApplication
   b. ServiceContractName: {http://contoso.com/hr/}IApplicationService
   c. Content: Click the ... button to display the content definition dialog and change the Message data to ApplicationRequest
d. Check the CanCreateInstance check box. This will cause every incoming message received by the Receive Job Application activity to create a new workflow instance.

![Properties of the ReceiveRequest activity](image1.png)

**Figure 19**

*Properties of the ReceiveRequest activity*

---

**Task 6 – Saving the Job Application**

You need to store the job application data in the HR database. The provided `SaveJobApplication` custom activity does this using the ADO.NET Entity Framework. When the job application is saved the database will generate an Application ID that you will use as the primary key for the application.

1. Drop a `SaveJobApplication` activity from the `HRApplicationServices.Activities` group inside the Save and Respond sequence above the `SendResponse` activity.
2. Set the properties of the `SaveJobApplicationActivity`
   
   a. AppRequest: `ApplicationRequest`
   
   b. Result: `ApplicationResponse`

**Task 7 – Configuring the Response**

Though you have stored the application in the database, you are not finished processing it. You want to let the user of the client application know that you have received and are processing their job application. And you have to initialize the correlation handle that you will use to identify messages bound for this workflow instance using content based correlation.

1. First, you need to create a correlation handle that will be used to correlate messages for this workflow instance. Select the Application Service scope, click the **Variables** button and just as you did before, add a new variable `ApplicationIDHandle` of type `CorrelationHandle` to the Application Service scope.

2. Next set the response data. In the **SendResponse** activity click on **View message...** to display the content definition dialog. Change the **Message** data to use the variable `ApplicationResponse`.

![Figure 21](image)

*Set the message data of the response to ApplicationResponse*

3. Now you can define the **CorrelationInitializers**. The initializer will extract data from the outgoing message using an XPath expression. The extracted key will be used to identify the persisted workflow instance. With the **SendResponse** activity selected, in the properties window click the button beside **CorrelationInitializers** to display the Add Correlation Initializers dialog.

4. Click **Add initializer** and add `ApplicationIDHandle`
5. With the ApplicationIDHandle initializer selected, in the XPath Queries window select the ApplicationID : Int32 message content

![Add Correlation Initializers](image)

Figure 22
Select the ApplicationID : Int32 message content

6. Click OK to dismiss the dialog.

7. The default namespace for types in your workflow service is tempuri.org. If you want to change the default namespace you will need to edit the XAMLX file using the XML editor. To do this right click on SubmitApplication.xamlx and select view code.

8. Locate the Name= attribute and insert p: in front of the name as shown. This will cause the service to use the namespace of the operation contract as the default namespace for the WSDL types

```xml
<WorkflowService
  mc:Ignorable="sap"
  ConfigurationName="SubmitApplication"
  sap:VirtualizedContainerService.HintSize="252,435"
  Name="p:SubmitApplication"
/>
```

Task 8 – Adding the Client Application

1. Add the existing HRClient project. To do this, in Solution Explorer right-click the solution node, and select Add / Existing Project. In the Open Project dialog, browse to Source\Assets folder of this lab (choosing the folder that matches the language of your preference) and select the HRClient project.

2. Add a Service Reference to the SubmitApplication.xamlx service, to do this
a. Right click on the HRClient project and select **Add Service Reference**

b. In the Add Service Reference dialog click on the **Discover** button to discover services in the solution. Visual Studio will find the `SubmitApplication.xaml` file. Double click it to see the details of the service. This will launch the service in the ASP.NET Development server.

c. Set the namespace of the service to **SubmitApp**

![Add Service Reference dialog](image1.png)

**Figure 23**

*Configuring the service reference*

d. Click the **Advanced** button, and check the **Generate asynchronous operations** checkbox.

![Service Reference Settings](image2.png)
Check the Generate asynchronous operations checkbox in the advanced settings.

3. Click **OK** to save and dismiss the Add Service Reference dialog.

4. From the menu select **Build / Build Solution** (Ctrl+Shift+B)

**Next Step**

**Exercise 1: Verification**

In this verification, you will run a client application verify that you have created a working service.

**Verification 1 – View the service in the browser**

1. Building the solution does not insure that your workflow is functional. The quickest way to determine if the workflow has any major problems is to view it in the browser. Right-Click **SubmitApplication.xaml** and select **view in browser**. You should see the Workflow Service.

   ![SubmitApplication Service](image)

   **Figure 25**
   
   *The Workflow Service*

   **Note:** If your workflow is invalid in some way you will see an ASP.NET Error page.

**Verification 2 - The client application can send and receive messages**

1. Right click the HRClient project, and select **Set as StartUp Project**.

2. Press **F5** to debug the application.

3. When the Candidate Information form comes up, press the **Submit** button
4. You should see the application being submitted.

5. Then you should see the response message from the server.
Next Step

Exercise 2: Email Notification

One of the advantages of Workflow Services is that they can do work beyond the initial request/response. In your workflow service you have completed only the first of three phases. In this exercise, you will implement the third phase of your workflow, where you will notify the client of the result by email. Later you will implement the second phase to screen the application.

Task 0 – Opening the Solution

To begin this exercise you can use the solution you finished from Exercise 1. Alternatively, you can follow the following steps to begin with Exercise 2.


   Open the starting solution for Exercise 2 located under the \Source\Ex2\Begin folder (choosing the folder that matches the language of your preference.) Use it as the starting point for this exercise.

2. Press CTRL+SHIFT+B to build the solution.

Task 1 – Modifying the Workflow Service
Your service has work to do after sending the initial response. In this task you send a hire notification email to the client.

1. In the HRApplicationService project, open the **SubmitApplication.xaml** file.
2. Select the **Application Service** sequence.
3. Add a Variable named **Hire** of type **Boolean** at the **Application Service** scope with a Default value of **False**.
4. To make it simple to send email, a **code activity** called **NotifyApplicant** is provided that will send an email. Drag a **NotifyApplicant** activity from the **HRApplicationServices.Activities** group, drop it in the **Notify Applicant** sequence and set the properties
   a. Display Name: **Notify Applicant Email**
   b. Hire: **Hire**
   c. Resume: **ApplicationRequest.Resume**

---

**Where will the email go?**

For testing purposes the web.config file is set to drop mail messages into a folder (C:\mailbox). You can view these email messages with the Outlook or the free Windows Live Mail client application.

---

**Next Step**

**Exercise 2: Verification**

In this verification, you will run the **HRClient** application to verify that it sends the hire notification email.

1. Right click the HRClient application, and select **Set as StartUp Project**.
2. Press **F5** to Debug the solution.
3. Click **Submit** to send the job application, you should see a thank you message as you did before.
4. Look in the C:\Mailbox folder. You should see an email message. If you have Outlook or Windows Live Mail client you can open it.
5. Open the message. The candidate will not be hired because the **Hire** variable defaults to **False**.
Exercise 3: Applicant Screening

Due to the high volume of résumés that Contoso receives, they want to automate a portion of the résumé screening process. In this exercise, you will send a message to a Flowchart Workflow Service that screens candidates based on education, and then you will notify the HR reviewer of applications that need review by a human.

Task 0 – Opening the Solution

To begin this exercise, you can use the solution you finished from Exercise 2. Alternatively, you can follow the following steps to begin with Exercise 3.

2. Open the starting solution for Exercise 3 located under the \Source\Ex3\Begin folder (choosing the folder that matches the language of your preference.) Use it as the starting point for this exercise.
3. Press CTRL+SHIFT+B to build the solution.

Task 1 – Adding the Education Screening Activity

In this task, you will add an existing activity that screens résumés based on the level of education.

1. Open ScreenEducation.xaml and explore it. This activity is implemented as a Flowchart that will be used to automatically screen out applicants based on the level of education. While the other activities in our HRApplicationServices.Activities library were implemented in code, this activity is implemented in XAML. You can create and share activities implemented in code or XAML using the Workflow Designer.
Figure 31
The ScreenEducation Flowchart

**Task 2 - Calling the ScreenEducation Activity**

1. Open `SubmitApplication.xaml`
2. Expand the `Screen Applicant` sequence
3. Add a new variable `EducationPassed` of type `Boolean` to the `Screen Applicant` scope. To do this
   a. Click the `Variables` button
   b. Click on `Create Variable`
   c. Name the variable `EducationPassed`
   d. In the type select `Boolean`
4. From the `HRAplicationServices.Activities` group drop a `ScreenEducation` activity onto the Screen Applicant sequence and set the properties
Task 3 – Conditionally Notifying a Human Reviewer

Not all applicants will pass the education screening. For those that do not, you need to update the database and notify them. For those that do pass you need to notify a human reviewer to check the application and make a decision.

1. Drop an If activity from the Control Flow group below the Auto Screen Education activity and set its properties
   a. Display Name: If Education Screen Passed
   b. Condition: EducationPassed

2. If the applicant failed the education screen you will simply update the database. Drop an UpdateHireApproved activity from the HRApplicationServices.Activities group on the Else part of the If activity and set the properties. The Hire variable defaulted to False so the applicant will not be hired in this branch.
   a. Display Name: Update No Hire
   b. ApplicantID: ApplicationResponse.ApplicationID
   c. HireApproved: Hire

3. If the applicant passed the education screen you need to notify a human reviewer and wait for a response. Drop a Sequence activity from the Control Flow group on the Then part of the If activity and set the properties.
   a. Display Name: Human Screening

4. You need to send an email to the HR Administrator to let them know there is an application to review. To do this drop the RequestHumanScreening activity from the HRApplicationServices.Activities group and set the properties.
   a. Display Name: Request Human Screening
   b. ApplicationID: ApplicationResponse.ApplicationID
   c. ApplicationRequest: ApplicationRequest

5. Now you need to wait for a response from the HR Administrator. They will be using a web page to send a message to your workflow with the result of the screening. Drop a TransactedReceiveScope activity from the Messaging group below the Request Human Screening activity and set the properties.
   a. Display Name: When Human Screening Complete
6. Add a variable HumanScreening of type HumanScreeningResult to the When Human Screening Complete scope.

7. Drop a Receive activity in the Request area and set the properties:
   a. Operation Name: HumanScreeningCompleted
   b. Service Contract Name: {http://contoso.com/hr/}IApplicationService
   c. Content: Click ![content icon] to display the Content Definition dialog and set the Message Data to HumanScreening

8. Because this message is sent to a workflow that already exists you must use correlation to provide a way for the WorkflowServiceHost to locate the correct workflow instance. To establish correlation for this Receive activity, do the following:
   a. CorrelatesOn: Click ![correlates on icon] to display the CorrelatesOn Definition dialog
   b. CorrelatesWith: ApplicationIDHandle
   c. Key: Select the AppID:Int32 value
   d. Click OK to dismiss the dialog

![CorrelatesOn Definition](image)

Figure 32
The Correlation definition for the Human Screening Receive activity

9. Once the message is received you need to update the database and send a reply. Since you have more than one activity in the body drop a Sequence and set the properties:
   a. Display Name: Update Human Screening Result

10. Now you can set the actual Hire value from the human screening result. Because Hire was initialized to False, the only way Hire is actually set to true is if the applicant passes both
automated and human screening. Drag an Assign activity from the Primitives group and drop it into the Update Human Screening sequence and set the properties

  a. Display Name: Assign Hire
  b. To: Hire
  c. Value: HumanScreening.HiringApproved

11. Drop an UpdateHireApproved activity from the HRApplicationServices.Activities group below the Assign Hire activity and set the properties.

  a. Display Name: Update Hire Result
  b. ApplicantID: ApplicationResponse.ApplicationID
  c. HireApproved: Hire

12. Now you need to send a response to the web application. To create a SendReply activity, right click the Receive activity, and select Create SendReply. Because the Receive activity is inside the Request area and the reply cannot be placed there the SendReply activity will be placed on the clipboard.

   Figure 33
   *Create a SendReply for the receive*

13. Paste the SendReply below the Update Hire Result activity and set its properties

  a. Display Name: Send Human Screening Reply
  b. Content - Message Data: True
14. From the menu select **Build / Build Solution** (**Ctrl+Shift+B**)

**Task 4 – Sending Human Screening Results**

The HireApproval.aspx page will need to call the SubmitApplication.xaml workflow. Even though they are in the same website the way you call into the workflow is by sending it a message. In this task, you will modify the page to send the hire message.
1. Next we need to Add a Service Reference to send a message to the receive activity of the SubmitApplication.xaml workflow that is waiting for the human screening result. To do this right click the HRApplicationServices project and select **Add Service Reference.**

2. In the Add Service Reference dialog click on Discover to find the service SubmitApplication.xaml.

3. Name the service reference **Application**, and click Ok to generate the reference.

4. In the **HRApplicationServices** project right click **HireApproval.aspx**, and select view code (F7).

5. Add the following namespace directives.

   *(Code Snippet - Introduction to WF Services Lab – HireApprovalNamespace CSharp)*

   **C#**
   
   ```csharp
   using HRApplicationServices.Contracts;
   using HRApplicationServices.Application;
   ```

   *(Code Snippet - Introduction to WF Services Lab – HireApprovalNamespace VB)*

   **Visual Basic**
   
   ```vb
   Imports HRApplicationServices.Contracts
   Imports HRApplicationServices.Application
   ```

6. Locate the **SendHumanScreenComplete** method, and modify the code as shown.

   *(Code Snippet - Introduction to WF Services Lab – SendHumanScreenComplete CSharp)*

   **C#**
   
   ```csharp
   private void SendHumanScreenComplete(bool hire)
   {
       ApplicationServiceClient proxy = new ApplicationServiceClient();
       try
       {
           // Create the result
           HumanScreeningResult result = new HumanScreeningResult()
           {
               AppID = Convert.ToInt32(LabelAppID.Text),
               HiringApproved = hire,
           };

           // Wrap it in a request
           HumanScreeningCompletedRequest request =
               new HumanScreeningCompletedRequest(result);

           // Call the workflow service
   ```
```vbnet
Private Sub SendHumanScreenComplete(ByVal hire As Boolean)
    Dim proxy As ApplicationServiceClient = New ApplicationServiceClient()
    Try
        Dim result As New HumanScreeningResult With
            { .AppID = Convert.ToInt32(LabelAppID.Text), .HiringApproved = hire
        }
        proxy.HumanScreeningCompleted(result)
        proxy.Close()
    Catch ex As Exception
        proxy.Abort()
        Throw
    End Try
End Sub
```

Next Step

**Exercise 3: Verification**

In this verification you will create a new application and verify the screening process.

1. Delete any mail messages from C:\Mailbox
2. Right click the **HRClient** project, and select **Set as StartUp Project**
3. Press **F5** to Debug the solution; the HRClient application should start
4. Submit a new application where the education level is **Bachelors**
5. The workflow should automatically reject this application and you should see a new email message in the C:\Mailbox folder. You can view the message if you like. Delete this message when finished.

6. In the Contoso HR Tool click Start Again. This time submit a new application where the education level is **Masters**

7. Look in C:\Mailbox. There should be a message, only this one is sent to the HR Reviewer. Right click and open the message.

![Screening Request for Applicant John Smith](image)

**Figure 35**
*Request for Application Review*

8. Click the link in the message to go to the HR Job Application Approval web page
9. Click **Hire** to hire the applicant. This will send a message to the workflow.

10. You should see a second mail message with the hire message.

### Next Step

**Exercise 4: Deploy the Solution**

In this exercise you will deploy, configure and monitor the solution using Windows Server AppFabric; a set of extensions to the Windows Application Server and the Windows Process Activation Service (WAS). The goal of Windows Server AppFabric is to simplify the deployment, configuration, management, and monitoring of Windows Communication Foundation (WCF) and Windows Workflow Foundation (WF) services built on .NET 4.

**Is AppFabric installed?**

AppFabric is **not installed** as a part of the .NET Framework 4 installation. It comes as a separate install delivered through a Windows Update package for Windows that you can install it using the **Microsoft Web Platform Installer** (recommended) or download **Windows Server App Fabric Beta 2** from the Download Center.
To determine if AppFabric is installed open IIS Manager. If AppFabric is installed when you open the Default Web Site you will see Windows Server AppFabric at the top of the features window.

Task 0 – Opening the Solution

To begin this exercise you can use the solution you finished from Exercise 3. Alternatively, you can follow the following steps to begin with Exercise 4.


2. Open the starting solution for Exercise 4 located under the \Source\Ex4\Begin folder (choosing the folder that matches the language of your preference.) Use it as the starting point for this exercise.

3. Press CTRL+SHIFT+B to build the solution.

Task 1 – Creating Configuration Transformations

Your application has several service references that were created in your development environment. When you deploy the application you need to transform the web.config file so that it uses the correct URIs from the production environment.

1. To create the configuration transformations in the HRApplicationServices project right click the web.config file, and select Add Config Transforms.

Figure 37
Right click web.config, and select Add Config Transforms in C#
Right click web.config, and select Add Config Transforms in Visual Basic

You should now see two additional config files under web.config

You should now have config transforms

Note: In Visual Basic these files are hidden by default. You must make them visible by enabling the “Show All Files” button in the Solution Explorer.

Open the Web.Release.config file. In this file you will add transformation instructions for how you want the web.config file to look after deployment. Modify it as shown


```xml
<?xml version="1.0"?>

<!-- For more information on using web.config transformation visit http://go.microsoft.com/fwlink/?LinkId=125889 -->

  <!--Modify the BaseURI key -->
  <appSettings>
    <add key="BaseURI" value="http://localhost/HRApplicationServices" xdt:Transform="SetAttributes(value)" xdt:Locator="Match(key)"/>
  </appSettings>
</configuration>
```
4. Switch the current configuration to **Release** to do a release build by selecting it from the toolbar.

![Switch to release build](image)

**Figure 40**

*Switch to release build*

---

**Task 2 – Creating the Deployment Package**

In this task you will use the Web Deployment feature inside Visual Studio to create a deployment package.

1. In the Solution Explorer right click the **HRApplicationServices** project, and select **Package/Publish Settings** set the options as follows

2. Items to deploy (applies to all deployment methods) section
   a. **Only files needed to run this application**
3. Web Deployment Package Settings section
   a. Create web package as a ZIP file
   b. Location: `obj\Debug\Package\HRApplicationServices.zip`
   c. IIS Web Site/Application name to be used on the destination server: **Default Web Site/HRApplicationServices**

4. Press **CTRL+SHIFT+S** to save all files – this will save your changes.

5. In the **Solution Explorer** right click the **HRApplicationServices** project, and select **Build Deployment Package**. You should see **Publish succeeded** as the status in the bottom left corner when the action has completed.

**Task 3 – Deploying the Package**

In this task, you will use the Web Deployment feature inside IIS Manager to deploy the package to the local web server.

1. Open **Internet Information Services (IIS) Manager** using the Start menu.

2. Expand the **Server** node, expand **Sites** and right click the **Default Web Site**. Select **Deploy / Import Application ...**
3. For **Package path** browse to the location where the package was created. By default this was in the `obj\Release\Package` folder for the `HRApplicationServices` project. Select `HRApplicationServices.zip`
4. Click **Next** 3 times to use the defaults on the rest of the Import Application Package wizard pages. When asked if you want to run this application in the default .Net 4 application pool click **Yes**.

5. Click **Finish** when the wizard has finished.

---

**Task 4 – Enabling the net.pipe Protocol on the Application**

AppFabric installs a Workflow Management Service (WMS) that works with the workflow persistence store to provide reliability and instance control. WMS communicates with the workflow service via the net.pipe protocol. Therefore, this protocol must be enabled on the application.

1. In **IIS Manager** right click on the **Default Web Site** and select the **Edit Bindings ...** option.

2. Ensure that **net.pipe** is added with * for its **Binding information**. If it is not, add it using the **Add...** button.

![Site Bindings](image)

**Figure 45**

*Enable the net.pipe site binding*

3. Click the **Close** button

4. Expand the Default Web Site and right click the **HRApplicationServices** application in the tree. Select **Manage Application / Advanced Settings**...

5. In the **Enabled Protocols** section after http add **,net.pipe** as shown
Enable the net.pipe protocol

Why enable the net.pipe protocol?

The net.pipe protocol must be enabled so that the Workflow Management Service can communicate with the Workflow Control standard endpoint of workflow services. This is how it can manage workflow services.

6. Click OK

Task 5 – Changing the Application Configuration

While you were developing the solution you were using the WF4Persistence database for persistence. In the IIS environment you are going to use a different persistence database. You also want to configure AppFabric to suspend workflows with unhandled exceptions.

1. In IIS Manager, expand the Default Web Site and right click the **HRApplciationServices** application, and select Manage WCF and WF Services / Configure...
2. Click the **Workflow Persistence** category.

3. Under SQL Server Workflow Persistence select **defaultSqlPersistenceStore**.

![Configure WCF and WF for Application](image)

**Figure 47**

*Configure persistence to use DefaultPersistenceConnectionString*

4. Click **OK** to dismiss the dialog. Click **Yes** when you are warned about recycling the application.

---

**Next Step**

**Exercise 4: Verification**

The Dashboard shows the activity for the services at a particular scope in the IIS tree. For example, if you click the server node at the top of the tree and then click the AppFabric Dashboard you will see activity
for all the WCF and WF services on the computer. In this verification, you will look at the Dashboard at your application level.

**Note:** To follow this verification, it is required to follow the steps in exercise 4. Opening the End solution will not be sufficient to follow the verification since there are some IIS configuration tweaks to perform.

1. **In IIS Manager**, expand the **Default Web Site**, and select the **HRApplicationServices** application in the tree.
2. At the bottom of the IIS Manager window, click **Content View** to see the content of the website.
3. Right click the **SubmitApplication.xaml** file, and select **Browse**.
4. You should see a browser window, with the **Submit Application Service**. If there are any problems with the configuration for the website this page will not appear.
5. At the bottom of the IIS Manager window click **Features View**.
6. Double click on the **AppFabric Dashboard** icon.

![HRApplicationServices Home]

Figure 48
*Open the Dashboard*

**What is on the Dashboard?**

**Persisted WF Instances:** This section shows the number of workflow instances that are live or in flight. These are the instances that have persisted to the database but have not yet completed.

**WCF Call History:** This section shows the number of WCF calls the have been received. The heading shows counts of the calls that were completed or that encountered exceptions.

**WF Instance History:** This section shows the number of workflow instances that have been activated, have failed, or completed.
7. Switch back to Visual Studio 2010

8. Open the app.config file in the HRClient project. Unlike the web.config which transformed when you deployed it, the app.config file must be manually edited. Change the endpoint address as shown in bold text below.

(Code Snippet - Introduction to WFServices Lab – WF4 ClientConfig XML)
9. Right click the HRClient project, and select Debug / Start new instance

10. In the Contoso HR Tool, set the Education to Masters, and click the Submit button. After a few moments you should see a confirmation message that the resume is being processed.

11. In IIS Manager, refresh the AppFabric Dashboard. You should see activity in each of the three sections.

12. Click on the 1 beside Active in the Persisted WF Instances section to see a list of the active persisted instances.

![Persisted WF Instances](image)

Figure 50
The SubmitApplication Workflow instance is active

**Note:** The host is configured by default to put active instances into the idle state after 60 seconds. If you do not have any instances in the Active state refresh the Dashboard and look for the instance in the Idle state.

13. Click on the hyperlink on the word Active or Idle in the Persisted WF Instances pane. This will open a query page with Persisted WF Instances

14. Click on the persisted instance. The Details pane at the bottom of the screen will show some information about the instance. In this example, you can see that the SubmitApplication Workflow has an active bookmark named HumanScreeningCompleted. That is the operation name of the bookmark created by the receive activity that it is currently waiting on.

**Note:** The active bookmark is visible only for instances that were idle
15. In the picture above you can see that there are a number of tracked events. To see these events, right click the instance, and select the View Tracked Events action. This action will take you to the Tracked Events page populated with all the monitoring information that was collected for the instance, sorted with the newest data at the top.
16. The last activity to execute was the **Request Human Screening** activity. Click on this row and notice in the **Details** pane at the bottom of the screen you will see information about this event. Its state is **Closed**, indicating that the **Request Human Screening** activity has finished executing.

17. In **Windows Explorer**, go to `C:\Mailbox` and find the latest message. Open it and click the link to review the application.

18. In the review web page, click the **Hire** or **No hire** button. This will send a message to the **SubmitApplication** workflow using the **HumanScreeningCompleted** operation.

19. Return to **IIS Manager** (still on the **Tracked Events** page), and click the **Run Query** button to refresh the data in the list (it may take up to a minute before the new events appear). You will now see that the remainder of the **SubmitApplication** workflow activities have executed.

20. Return to the **Dashboard** and refresh it. You should now see no active or idle instances in the **Persisted WF Instances** section. In the **WF Instance History** section you should see that 2 workflow instances have been activated and completed first the **ScreenEducation** and then the **SubmitApplication** workflow.
The Dashboard showing 2 completed workflows.

Next Step

Exercise 5: Troubleshooting

Exercise 5: Troubleshooting

What happens when a workflow encounters an error? If the workflow is responding to a web service message it can send a SOAP fault to let the caller know there was an error. But if the workflow is doing work after it sent a response you need a way to find out what went wrong so you can take action. What would happen if your SubmitApplication workflow was unable to send the Request Human Screening email? In this exercise you will force an error that simulates a mail failure then use AppFabric to troubleshoot and correct the problem.

Task 0 – Opening the Solution

To begin this exercise you can use the solution you finished from Exercise 4. Alternatively, you can follow the following steps to begin with Exercise 5.


2. Open the starting solution for Exercise 5 located under the Source\Ex5\Begin folder (choosing the folder that matches the language of your preference.) Use it as the starting point for this exercise.

3. Press CTRL+SHIFT+B to build the solution.

Task 1 – Simulating an Error Condition

1. Using Windows Explorer Rename the C:\Mailbox folder to C:\Mailbox2 and delete any mail messages in the folder. When the workflow tries to send the email it will now fail at this step.
2. Switch back to **Visual Studio 2010** right click on the **HRClient** project and select **Set as StartUp Project**

3. Press **Ctrl+F5** to start the **HRClient** project without debugging

4. In the Contoso HR Tool set the **Education** to **Masters**, and click the **Submit** button. After a few moments you should see a confirmation message that the resume is being processed. From the client perspective it appears that everything is working.

5. In **IIS Manager** expand the **Default Web Site**, and select the **HRApplicationServices** application in the tree. Double click the **AppFabric Dashboard** icon.

6. In the **Persisted WF Instances** section at the top of the Dashboard you should see 1 instance in the **Suspended** state. Click the **1** to go to the **Persisted WF Instances** page.

   ![Suspended: 1](image1.png)

   **Figure 54**
   
   You should see a suspended instance

7. Select the one suspended instance in the list, and look at the **Details** pane at the bottom.

8. Click the **Errors** tab. You should see **Failure sending mail**.

   ![Errors tab](image2.png)

   **Figure 55**

   Errors tab explains why the workflow was suspended

9. At this point, you know the problem has something to do with a failure to send email. To get more information about the error right click the suspended instance, and select the **View Tracked Events** action.
Figure 56
To get more detail, view tracked events

10. You will see some **warning** and **error** events in the list. Click on these, and then the **Errors** tab to see more information about the error.
The WorkflowInstanceAbort record showing the cause of the failure

11. You can see from the error that the mail system could not find the path C:\Mailbox because you renamed it earlier to c:\Mailbox2.

Task 2 – Correcting the Error and Resuming the Workflow

1. Switch back to Windows Explorer, and rename the Mailbox2 folder to Mailbox.
2. Switch to the **IIS Manager**, and click the back button to return to the **Persisted WF Instances** page.

**Figure 58**
*Click the back button to return to the Persisted WF Instances page*

3. Right click the suspended instance, and select the **Resume** action. This will queue the command to resume the instance from its last persist point.
4. Wait a few seconds, and click the Run Query button. You should see an empty list because the workflow resumed and completed.

5. Return to the Dashboard and refresh it. You should now see 1 persisted WF instance in the Active or Idle state depending upon how long it takes you to view it.

6. In Windows Explorer go to C:-Mailbox and find the latest message. Open it, and click the link to review the application.

7. In the review web page click the **Hire** or **No hire** button. This will send a message to the SubmitApplication workflow using the HumanScreeningCompleted operation.

8. In the C-Mailbox folder you should see another message with the result

9. Switch back to IIS Manager, and refresh the Dashboard. You should see the Completions count incremented by 1. Click **Completions**: you will be able to review the statistics about the Submit Application workflow as shown.

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Machine</th>
<th>Site Name</th>
<th>Service...</th>
<th>Status</th>
<th>Creation Time</th>
<th>Completion Time</th>
<th>Last Aborted Time</th>
<th>Lifetime Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>EducationScreening</td>
<td>mjecker1</td>
<td>DefaultWeb</td>
<td>n/afug...</td>
<td>Complete..</td>
<td>10/23/2009 1:03:41.761 PM</td>
<td>10/23/2009 1:03:41.761 PM</td>
<td></td>
<td>00:00:00:000000</td>
</tr>
<tr>
<td>EducationScreening</td>
<td>mjecker1</td>
<td>DefaultWeb</td>
<td>n/afug...</td>
<td>Complete..</td>
<td>10/23/2009 1:03:41.761 PM</td>
<td>10/23/2009 1:03:41.761 PM</td>
<td></td>
<td>00:00:00:000000</td>
</tr>
<tr>
<td>SubmitApplication</td>
<td>mjecker1</td>
<td>DefaultWeb</td>
<td>n/afug...</td>
<td>Complete..</td>
<td>10/23/2009 1:03:41.761 PM</td>
<td>10/23/2009 1:03:41.761 PM</td>
<td></td>
<td>00:00:00:000000</td>
</tr>
<tr>
<td>SubmitApplication</td>
<td>mjecker1</td>
<td>DefaultWeb</td>
<td>n/afug...</td>
<td>Complete..</td>
<td>10/23/2009 1:03:41.761 PM</td>
<td>10/23/2009 1:03:41.761 PM</td>
<td></td>
<td>00:00:00:000000</td>
</tr>
</tbody>
</table>

**Figure 60**

*The completed workflow*

**Next Step**

**Summary**

In this lab you have learned how to build, send and receive messages using Workflow Services. You’ve also seen how you can deploy, monitor and troubleshoot services using AppFabric.
What do you think of this lab? What do you think of the new Windows Workflow 4? Your feedback is important; it will help us build the best product for you. Please take a moment to provide it. Send your comments to wfwcfhol@microsoft.com