3. The design view for flowcharts looks slightly different than sequential workflows. The green circle indicates where the workflow starts. We need to create a new activity to read input from the user. Create a new class called ReadInput.

4. Enter the following using statement:
   ```csharp
   using System.Activities;
   ```

5. Now enter the following code:
   ```csharp
   public class ReadInput : CodeActivity<Int32>
   {
   protected override Int32 Execute(CodeActivityContext context)
   {
   return Convert.ToInt32(Console.ReadLine());
   }
   }
   ```

6. Save the class and compile the application.

7. Open Workflow1.xaml.

8. Drag a WriteLine activity beneath the green circle and change the Display Name to “What is your age?” and set the Text to “What is your age?”
9. Drag the new ReadInput activity beneath the “What is your age?” activity and change the display name to “Read input.”

10. Create a new variable called age of type Int32.

11. On the ReadInput activity, set the Result property to age.

The next thing to do is determine if the customer is old enough to see the film (which in this case will always have an 18 rating). Flow chart workflows have a new type of activity not found in sequential workflows, called FlowDecision.

1. Drag a FlowDecision activity beneath the read input block and change the condition to Age >= 18. There are obviously two possibilities to this expression:
   • Customer is old enough so they can see the film (FlowDecision condition = true).
   • Customer is too young, so shouldn’t be seeing any movies (FlowDecision condition = false).

2. To simulate the customer failing age verification, drag a WriteLine activity to the right of the flow decision and change the display name and text to “Sorry not old enough.”

3. Drag another WriteLine activity beneath the flow decision and change the display name and text to “Age validation successful.”

4. We now need to link up the activities we have just created. Move the mouse over the green circle that indicates the start of the flow chart workflow, and three grey dots will appear around it. Click the one on the bottom of the circle and then drag the mouse down to the ReadInput activity.

5. When you near the WriteLine activity, three grey dots will appear around it. Drag the line to one of these dots and then release the mouse button to link up the start of the workflow with our read line activity.

6. Link up the “What is your age?” and ReadInput activities.

7. We need to join the FlowDecision up to the workflow. FlowDecision activities have two nodes, true or false, that surprisingly indicate the path to take when the condition specified is true or false. Drag the false node to the “Sorry not old enough” WriteLine activity and then drag another line from “Sorry not old enough” back round to the ReadInput activity.

8. Drag the true node on the FlowDecision activity to the “Age validation successful” activity.

9. Finally drag a line between the “What is your age?” and ReadInput activity. Your final workflow should look like Figure 6-14.

10. Open Program.cs and add a Console.ReadKey(); beneath the invoke command so the application doesn’t close immediately.

11. That’s it; your workflow is ready to run. Press F5 to run it.

12. Try entering different ages and note that unless you enter at least 18 the workflow will write “Sorry not old enough.”
WCF/Messaging Improvements

A number of enhancements have been introduced in WF4 to improve integration with WCF and to ease messaging scenarios.

Correlation

Correlation functionality first appeared in WF3.5 and allows you to route incoming messages to specific workflow instances based on their content or protocol used. For example if you have a very long running workflow where replies take weeks or months to return it is important that when a reply is received it is sent to the correct individual workflow.

ReceiveAndSendReply and SendAndReceiveReply are the new activities discussed in the following sections that provide a correlated send and receive activities with a number of new methods of correlation such as xpath and correlation scope.

WCF Workflow Service Applications

WCF Workflow Service applications are a new type of project in VS2010 that make it very easy to create workflows for sending and receiving data. They essentially provide a declarative WCF service defined
using workflow activities. WCF Workflow Service applications have all the benefits of WF such as support for long-running services, GUI interface, and also the additional benefits that as they are declared declaratively so are easy to deploy and version.

VS2010 comes with a WCF Workflow Service Application template that you can adapt for your own needs. The sample application simply echoes a number you send to it back to you. Let’s take this for a spin now.

1. Create a new WCF Workflow Service project called Chapter6.WFService. The template will contain a sequential activity looking very similar to Figure 6-15.

2. This sequential activity is defined in the file Service1.xaml. If you open this up with the XML editor you will see the XAML that defines this service (boring bits removed as it’s pretty long):

```xml
<Sequence DisplayName="Sequential Service"
    sad:XamlDebuggerXmlReader.FileName="D:\wwwroot\book\Chapter6_WF\Chapter6.WFService\Chapter6.WFService\Service1.xaml">
    <Variables>
        <Variable x:TypeArguments="CorrelationHandle" Name="handle"/>
        <Variable x:TypeArguments="int" Name="data"/>
    </Variables>
    ...
</Sequence>
```

Figure 6-15. WF Service project
<Receive x:Name="__ReferenceID0" DisplayName="ReceiveRequest" OperationName="GetData" ServiceContractName="contract:IService" CanCreateInstance="True">
  <Receive.CorrelationInitializers>
    <RequestReplyCorrelationInitializer CorrelationHandle="[handle]" />  
  </Receive.CorrelationInitializers>
  <ReceiveMessageContent>
    <p:OutArgument x:TypeArguments="x:Int32">[data]</p:OutArgument>
  </ReceiveMessageContent>
</Receive>

<SendReply Request="{x:Reference Name=__ReferenceID0}" DisplayName="SendResponse">
  <SendMessageContent>
    <p:InArgument x:TypeArguments="x:String">[data.ToString()]</p:InArgument>
  </SendMessageContent>
</SendReply>

</p:Sequence>
</WorkflowService>

3. As the template service doesn’t do anything apart from echo a value back, we are going to modify it slightly so we can see a change. In the SendResponse box click the Content text box and amend the Message data property to the following:
   data.ToString() + " sent from WF service"

4. Click OK.

5. Save the project.

6. Now add a new console application to the solution called Chapter6.WFServiceClient.

7. Add a service reference to Chapter6.WFService (click Add Service Reference then Discover►Services in Solution; it will be listed as Service1.xaml).

8. Leave the namespace as ServiceReference1.

9. In Chapter6.WFServiceClient modify Program.cs to the following:
   Console.WriteLine(client.GetData(777));
   Console.ReadKey();

10. Set Chapter6.WFServiceClient as the startup project and press F5 to run. You should see the message “777 sent from WF Service” output to the console.

   If you wanted to deploy this service, you could simply copy the Service1.xaml and Web.config file to a web server or even host it using “Dublin.”

Activities

A number of new activities are introduced in WF4, and some activities from WF3 have been dropped. Note the Microsoft upgrade documents mentioned at the start of this chapter contain more detail on these changes and suggest an upgrade path.
WF3 Activity Replacements
Some existing WF3 activities have now been dropped. The suggested replacements are listed below:

- IfElse becomes If or Switch.
- Listen becomes Pick.
- Replicator becomes ForEach or ParallelForEach.
- CodeActivity is gone and you should use activity customization as described above.

New Activities
WF4 introduces a number of new activities.

AddToCollection, RemoveFromCollection, ExistsInCollection & ClearCollection
Activities for working with collections in your workflows.

Assign
Assign allows us to assign values to variables and arguments and has been used extensively in the previous examples.

CancellationScope
CancellationScope allows you to specify activities to be run should an activity be cancelled. The body section surrounds the code you may wish to cancel and the cancellation handler section specifies code to run if an activity is cancelled. See Figure 6-16.

Figure 6-16. CancellationScope
CompensatableActivity
An advanced activity used for long running workflows that allows you to define compensation, confirmation, and cancellation handlers for an activity. This is used in conjunction with the compensate and confirm activities.

DoWhile
DoWhile continues to run code until the condition specified is true. The code inside it will be run at least once. See Figure 6-17.

![Figure 6-17. DoWhile](image)

ForEach
An activity for looping round a collection.

Interop
Interop allows you to use your existing WF3 activities and workflow in a WF4 application. Interop can wrap any non-abstract types inherited from System.Workflow.ComponentModel.Activity. Any properties are exposed as arguments to WF4. Interop can help migration from WF3 or allow you to use existing WF3 workflows you don’t possess the source to.

InvokeMethod
InvokeMethod allows the calling of an existing static method. You can pass generic types, pass parameters by reference and also call it asynchronously.
Parallel
Parallel activity was present in WF3, but didn’t truly run activities in parallel (it used time slicing). In WF4 the Parallel activity and ParallelForEach the activities now run truly in parallel subject to suitable hardware.

Persist
Persist allows you to persist the workflow instance using the current workflow configuration settings. You can also specify areas where state should not be persisted with no-persist zones.

Pick
Provides functionality for using events and replaces WF3s listen activity.

ReceiveAndSendReply and SendAndReceiveReply
WF4 has improved support for messaging scenarios by introducing a number of new activities for sending and receiving data between applications and improved support for correlating messages. WF4 introduces the ReceiveAndSendReply (Figure 6-18) and SendAndReceiveReply (correlated versions of Send and Receive) activities that allow you to specify code to run in between the initial Send or receive.

Figure 6-18. ReceiveAndSendReply
These are advanced activities so please see the WF SDK for an example of their usage. The messaging activities can operate with the following types of data:

- Message
- MessageContracts
- DataContracts
- XmlSerializable

**TryCatch**

TryCatch allows you to specify activities to be performed should an exception occur and code that should always run in a Finally block similar to C# or VB.NET. See Figure 6-19.

![Figure 6-19. TryCatch](image)

**Switch<T> and FlowSwitch**

Switch is similar to the switch statement in C# and contains an expression and a set of cases to process. FlowSwitch is the flowchart equivalent of the switch statement. See Figure 6-20.
Powershell and Sharepoint Activities

In the preview versions of WF4, you may have seen Powershell, Sharepoint, and Data activities. These are now moved to the workflow SDK because Microsoft has a sensible rule for framework development that .NET should not have any dependencies on external technologies.

Misc Improvements

WF 4 also contains a number of other enhancements.

- WF version 4 is 10 to 100 times faster than WF3, and performance of the WF designer is much improved.
- Programming model has been simplified.
- Expressions now have intellisense (which can be utilized by activities you create as well).
- WF4 has support for running workflows in partial trust environments.
- Improved support for declarative (XAML only) workflows.
- Add breakpoints can be added to XAML directly. If breakpoints are added in design view and you switch to XAML view they will be shown (and vice versa). See Figure 6-21.