aborting a workflow
An ungraceful and immediate teardown of a workflow instance. An aborted workflow that has been previously persisted can be loaded and resumed from its last persistence point. Aborting the workflow simply throws away everything since the last persistence point.

activity
The basic unit of composition within a workflow. An activity represents a single step in a tree of declared objects. Each activity can have its own set of input and output arguments and local variables and accomplishes some useful work.

Activity
The abstract base class for all other activity types in WF.

ActivityAction
A type-safe activity callback mechanism that allows you to pass 0–16 arguments with no return value. The target activity that is executed must match the argument signature of the ActivityAction. This class is derived from ActivityDelegate.

ActivityBuilder
A class that is designed to assist with the construction of activities. It enables the workflow designer to support the declaration of arguments and variables.

ActivityContext
A class that provides an activity with limited access to the WF runtime environment. The runtime environment includes the variables and arguments that are currently in scope for a single execution of an activity. This is the base class for the CodeActivityContext and NativeActivityContext classes that provide varying levels of access to the runtime environment.
**ActivityDelegate**

Represents a callback mechanism using an activity. This type allows you to define a typesafe callback for an activity. The callback activity to execute is specified at design time when the activity is used. This is the parent class of ActivityAction and ActivityFunc.

**ActivityDesigner**

A class that visually represents an activity on the workflow designer surface. You must derive from this class when you implement a custom designer for an activity.

**ActivityFunc**

A type-safe activity callback mechanism that allows you to pass 0–16 arguments with a return value. The ActivityFunc class is similar to the ActivityAction class except it supports a return value while ActivityAction does not. This class is derived from ActivityDelegate.

**ActivityInstance**

A class that represents the current runtime state of an activity.

**arguments**

Define the public inputs and outputs of an activity. The Argument class is the base for all arguments. Classes derived from this base (InArgument, OutArgument, InOutArgument) define and restrict the direction of data flowing into and out of an activity.

**AsyncCodeActivity**

An abstract base class that is used to implement an activity that performs work on a separate thread. This class is derived from the base Activity class.

**asynchronous unit of work**

A general category of work that an activity might perform. Activities in this category are designed to perform an atomic unit of work but are capable of doing so asynchronously on a separate thread. Although these activities can use a separate thread for part of their work, that work must execute in a relatively short period of time. They are not designed for long-running operations that might take days to complete. The work they perform should be atomic, completing entirely during a single execution of the activity.

**atomic unit of work**

A general category of work that an activity might perform. This kind of activity encapsulates the logic to perform a unit of work synchronously on the workflow thread. The unit of work that is performed is atomic in the sense that it is completed entirely during a single execution of the activity. It doesn’t need to suspend execution and wait for external input. It is short-lived and doesn’t perform time-consuming operations. It executes synchronously on the workflow thread and doesn’t create or use other threads.
binding
In WCF, a binding describes how messages are exchanged between systems using an endpoint. It is a named stack containing the transport, encoding, and protocols that determine the way that WCF processes inbound and outbound messages on an endpoint.

bookmark
A general mechanism to temporarily suspend execution of an activity until external input is received. When the external input is available, the bookmark is resumed. Bookmarks can also be configured to not block execution of an activity and to be resumed multiple times.

CacheMetadata
A method of the Activity class that creates and validates a full description of the activity. The description includes the arguments, variables, child activities, child delegates, and other implementation details.

canceled activity state
The state that an activity transitions to when cancellation has been requested.

canceling a workflow
The most graceful way to stop execution of an instance. Canceling a workflow triggers any optional cancellation and compensation logic that you have declared within the workflow. It leaves the workflow in the canceled state. A canceled workflow cannot be resumed.

closed activity state
The state that an activity transitions to when it has successfully completed its work. An activity in the closed state cannot be resumed.

CLR
See Common Language Runtime.

CodeActivity
An abstract base class that is used to implement a custom activity. The work that is performed within the custom activity should be short-lived and execute synchronously on the workflow thread. This class is derived from the base Activity class.

Common Language Runtime
The managed runtime environment that permits cross-language .NET development.
compensation
The general mechanism by which work that has already been completed is undone.

composite activities
Activities that are designed to support one or more child activities or activity delegates.

composition
A way to implement new activities by combining existing child activities to serve a new specialized purpose.

confirmation
A mechanism to signal that an activity is past the point of execution where compensation is allowed or necessary.

constraints
Custom classes that externally perform validation on an activity.

content-based correlation
Correlation that is based on one or more selectable data elements within a message. This type of correlation uses application data to correlate messages rather than data that is provided by the infrastructure.

context-based correlation
Correlation that is based on fixed and well-known headers in the message that are provided by the infrastructure.

control flow activities
A general category of activities that allow you to make branching and looping decisions with a workflow.

correlation
A mechanism used to associate messages with the correct workflow service instance. Correlation is also used to associate multiple messaging activities with each other within a workflow.

data contract
In WCF, a data contract is one that permits the definition of messages with multiple parameters.
declarative model
An implementation model that defines the work to be performed using a tree of activities. The relationships between the activities determines the sequence and conditions under which each activity is executed. In a declarative model, imperative code is not used to control the flow of execute between activities.

duplex message exchange pattern
One of the WCF message exchange patterns supported by WF. When using this pattern, each side of the conversation can send out-of-band messages directly to the other side.

durable delay
Refers to a declared delay in a workflow that is persisted to storage during the delay period. When the delay expires, the workflow is reloaded from storage, and processing can continue. The delay is durable in the sense that it is not lost when the workflow is unloaded from memory.

DynamicActivity
An activity class that is used to dynamically construct activities at runtime rather than at design time. A DynamicActivity cannot be compiled into a CLR type. This class is derived from the base Activity class.

EditingContext
A class that is used by the WorkflowDesigner to manage internal designer state. Managed state includes services and context items.

endpoint
In WCF, a place where messages can be sent or received. All messages in WCF are exchanged between endpoints.

ETW
See Event Tracing for Windows.

Event Tracing for Windows
An enhanced Windows tracing facility that provides greater management and viewing of trace events.

executing activity state
The state that all activities begin in. This state indicates that an activity has not yet completed its work. An activity remains in the executing state when it becomes idle.
**expressions**
A general-purpose mechanism to declaratively execute some logic and return a value. Expressions in WF include any activities that return a single value. Expressions allow you to declare code that assigns a value to a property, saves a value to a workflow variable, and so on. Expressions are the small bits of glue code within the declarative workflow model.

**extension**
See workflow extension.

**external data exchange**
A WF 3.x mechanism that is used to facilitate communication between the host application and a workflow instance.

**faulted activity state**
The state that an activity transitions to when an unhandled exception has been thrown or when the activity has been terminated. An activity in the faulted state cannot be resumed.

**flow chart modeling style**
A style of workflow authoring that uses direct connections between activities to control the flow of execution. Looping and branching decisions are made using activity connections. This is in contrast with the procedural modeling style that adds activities as children of flow-control activities to control the flow of execution.

**Flowchart**
An activity that defines the flow chart modeling style. Child activities can be added to this activity, and direct connections can be defined between the activities to control the flow of execution.

**FlowDecision**
An activity that can be used as a child of the Flowchart activity to make branching decisions. This activity enables branching of the flow of execution based on the result of a Boolean condition.

**FlowSwitch<T>**
An activity that can be used as a child of the Flowchart activity to make branching decisions based on the value of an expression. A different flow of execution can be defined for each possible value of the expression.
**human interaction**
A general category of workflows that require interaction with humans. Workflows of this type are typically capable of long-running execution because of the unpredictability of the human interactions.

**instance store**
The persistence mechanism that is used by WF 4. Regardless of the actual persistence medium, all persistence is accomplished using a component known as an *instance store*.

**InstanceStore**
The base class that all instance store components must derive from.

**Interop**
An activity that is capable of wrapping an existing WF 3.x activity for execution in the WF 4 runtime environment.

**Lambda expressions**
An expression syntax that is used to define and execute anonymous functions.

**Language Integrated Query**
A set of extensions to the .NET Framework that provide native language support for queries. Support is currently available for C# and Visual Basic.

**LINQ**
See Language Integrated Query.

**LINQ to SQL**
LINQ extensions that provide the ability to query and manage data in a SQL database.

**long-running unit of work**
A general category of work that an activity might perform. This kind of activity is designed to perform work that may take a very long time to complete. It may perform part of its work and then wait until some external input is received. While it is waiting, the activity is idle and is not occupying the workflow thread. WF supports this long-running behavior using a concept called *bookmarks*.

**MEP**
See message exchange pattern.
**message contract**
In WCF, a message contract is one that supports only a single argument with a high degree of control over the shape of the actual message.

**message exchange pattern**
An established pattern for the exchange of messages between applications. Well-known patterns include One-way, Request/Response, and Duplex.

**MetadataStore**
A workflow designer runtime component that maps activity types to the designers that will be used for each activity.

**modeling style**
Describes the design metaphors that are used when you declare an activity or workflow. Two modeling styles that are available with WF are the procedural and flowchart modeling styles.

**ModelItem**
In the workflow designer, a ModelItem represents a single activity that is capable of being maintained. The workflow designer represents the activities in a design as a tree of ModelItem objects.

**NativeActivity**
An abstract base activity class that permits full and unrestricted access to all available functionality of the WF runtime. Included is the ability to schedule the execution of other activities and delegates and to create and resume bookmarks. This class is derived from the base Activity class.

**one-way message exchange pattern**
One of the message exchange patterns supported by WF. When using this pattern, messages are sent to an endpoint without any expectation of a response. One-way messages are also known as fire-and-forget messages since there is no direct confirmation that the message was received and acted upon.

**parallel execution**
Describes the ability of WF to interleave the execution of multiple scheduled activities.
persistence
- The ability to save the current state of an activity or workflow to a durable store and reload it for execution at a later time. Activities are typically persisted when they are idle and waiting for external input. The activity is reloaded from the durable store when the external input is available.

PersistenceParticipant
- The base class for any classes that extend the standard persistence mechanism. Classes that derive from this base class can participate in persistence by adding additional data elements to be saved and loaded.

procedural modeling style
- A style of workflow authoring that uses familiar flow-control constructs such as if, while, and switch to control the flow of execution between activities. In this modeling style, the activities to execute are added as children of the flow-control activity.

request/response message exchange pattern
- One of the message exchange patterns supported by WF. When using this pattern, messages are organized into pairs. After sending a request message to an endpoint, the requesting application waits for the receipt of a response message.

rules
- In WF 3.x, rules are statements of fact concerning some data. WF 3.x includes a general rules engine that allows you to execute multiple rules (see RuleSet) against data.

RuleSet
- In WF 3.x, this class defines multiple rules that are designed to be applied to data as a single set.

scheduling
- The mechanism by which activities are executed. Child activities are not directly executed by their parent activity. Their execution is scheduled with the workflow runtime. The workflow runtime manages a queue of work items to determine the sequence in which activities are executed.

state machine
- A WF 3.x modeling style that defines a workflow as a set of states and possible events. The events define interactions that the workflow is capable of handling while it is in each state. Transitions between states are also modeled.
system interaction
A general category of workflow that interacts with other parts of the application and does not require human interaction.

terminating a workflow
 Stops execution of a workflow but does not trigger cancellation and compensation logic. The workflow instance is left in the faulted state and cannot be resumed.

tracking
A mechanism that enables instrumentation of workflows. Detailed tracking data is available for each activity in a workflow.

tracking participant
A component that is registered with the tracking system and receives a flow of tracking records.

tracking profile
A configurable profile that filters the tracking records that are passed to each tracking participant.

tracking record
The raw tracking data that is produced by the tracking system, filtered by a tracking profile, and consumed by one or more tracking participants.

transaction
A construct that guarantees the atomicity and integrity of work that is performed, especially to a database.

validation
In WF, validation ensures the correct usage of an activity. Arguments may be validated for correctness, and the relationships between the activity and other activities in the model may be examined.

variable
Defines a named storage location within an activity. Variables are typically used for the storage of intermediate results within a workflow. Variable values are scoped to a single execution of an activity.
variable scoping
Describes the characteristic of variables that limits their visibility and access to the activity that defined them and any child activities.

VB Expressions
Expressions that are implemented using Visual Basic syntax. See expressions.

WCF
See Windows Communication Foundation.

WF
See Windows Workflow Foundation.

WF 3.x
Refers to versions 3.0 and 3.5 of the Windows Workflow Foundation. This version of WF is not covered in this book.

WF 4
Refers to version 4.0 of the Windows Workflow Foundation and the subject of this book.

Windows Communication Foundation
A framework that is provided with the .NET Framework to enable message-based communication within or between applications.

Windows Workflow Foundation
A framework that is provided with the .NET Framework to enable developers to implement workflow-based applications.

workflow
An ordered series of steps that accomplish some defined purpose according to a set of rules.

workflow extension
A class that can provide functionality for one or more workflow instances and the host application. A custom workflow extension can provide any functionality that is required by your application. A single thread-safe instance of an extension can be shared by multiple workflow instances, or a new instance of an extension can be created for each workflow instance.
**Workflow services**

WCF services that are implemented as a WF 4 workflow.

**WorkflowApplication**

A class that allows you to execute an activity with complete access to the features of the WF runtime. This class includes the ability to use features such as persistence, extensions, and bookmarks.

**WorkflowDesigner**

The class that encapsulates the workflow designer functionality. This is the primary class that enables rehosting of the workflow designer in your own application.

**WorkflowInvoker**

A class that provides a simplified way to execute an activity as if it were a method. The class provides limited runtime functionality.

**WorkflowServiceHost**

The class that enables self-hosting of WCF-based workflow services within your own application.

**Xaml**

A declarative XML-based markup language that is used by WF to declare activities and workflows.

**Xamlx**

A type of file that contains a Xaml declaration of a workflow service.
Comparing WF 3.x to WF 4

WF 3.x to WF 4 Architectural Differences

The following table summarizes the major architectural differences between WF 3.x and WF 4.

<table>
<thead>
<tr>
<th>Category</th>
<th>WF 3.x</th>
<th>WF 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authoring styles</td>
<td>WF 3.x supports sequential or state machine workflows, but the two styles cannot be mixed. If you start with a sequential workflow, you can’t mix state machine activities in the same workflow.</td>
<td>WF 4 supports procedural and flowchart modeling styles out of the box. Other modeling styles (such as state machine) may be provided in the future. Modeling styles are determined by the individual activities and are not restrictive based on the starting workflow type. Styles can be mixed within a single activity or workflow.</td>
</tr>
<tr>
<td>Code-beside</td>
<td>WF 3.x requires a code-beside file for property and variable definition and for event handling.</td>
<td>WF 4 does not require or support a code-beside file. Variables and arguments are fully declarative. Activity delegates can be used to execute callback functionality.</td>
</tr>
<tr>
<td>Conditions</td>
<td>WF 3.x supports Boolean conditions that are implemented in code or as declarative rule conditions.</td>
<td>WF 4 uses expressions to define Boolean conditions.</td>
</tr>
<tr>
<td>Confirmation</td>
<td>WF 3.x supports compensation but does not support confirmation.</td>
<td>WF 4 supports compensation and also supports confirmation. Confirmation is used to indicate that the workflow has progressed past the point where compensation is required or available as an option.</td>
</tr>
</tbody>
</table>
## Comparing WF 3.x to WF 4

### Continued

<table>
<thead>
<tr>
<th>Category</th>
<th>WF 3.x</th>
<th>WF 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirmation</td>
<td>WF 3.x supports compensation but does not support confirmation.</td>
<td>WF 4 supports compensation and also supports confirmation. Confirmation is used to indicate that the workflow has progressed past the point where compensation is required or available as an option.</td>
</tr>
<tr>
<td>Data model</td>
<td>WF 3.x manages data within an activity or workflow using private variables and public properties. The properties can be CLR types or dependency properties. Individual public property values can be used as input or output parameters or as temporary storage within a workflow.</td>
<td>WF 4 introduces an explicit data model that uses variables for storage within an activity and arguments for parameters.</td>
</tr>
<tr>
<td>Dynamic updates</td>
<td>WF 3.x supports the ability to dynamically alter the definition of a workflow after it has started execution.</td>
<td>WF 4 does not support dynamic updates to executing workflows.</td>
</tr>
<tr>
<td>Executing workflows</td>
<td>Workflows are executed using the WorkflowRuntime and WorkflowInstance classes.</td>
<td>The WorkflowApplication class is used to execute a workflow and provides access to all features of the workflow runtime. The WorkflowInvoker class provides a much simplified way to execute workflows with limited access to runtime features.</td>
</tr>
<tr>
<td>File Types</td>
<td>WF 3.x uses Xoml files for code separation activities and workflows. The Xoml file contains the declarative portion of the workflow and a separate code-beside file contains properties, variables, and event handler code.</td>
<td>WF 4 uses Xaml files for declarative activities and workflows. The Xaml file contains the complete declaration of an activity, so a separate code-beside file is not required. Xamlx files are used to declare workflow services.</td>
</tr>
<tr>
<td>Flowchart</td>
<td>WF 3.x does not support a free-form flowchart modeling style. Workflow types supported in WF 3.x include sequential and state machine.</td>
<td>WF 4 supports a Flowchart activity that allows free-form organization of the activity tree. The flow of control is managed using direct links between activities.</td>
</tr>
<tr>
<td>Flowed transactions</td>
<td>WF 3.x does not support flowing an external transaction into a workflow.</td>
<td>WF 4 supports flowing an external transaction into a workflow.</td>
</tr>
<tr>
<td>Category</td>
<td>WF 3.x</td>
<td>WF 4</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td>Host communication</td>
<td>Workflow queues are used for communication between the host application and a workflow instance.</td>
<td>Bookmarks are used for communication between the host application and a workflow instance.</td>
</tr>
<tr>
<td>Parameter direction</td>
<td>In WF 3.x, parameters are defined as public properties. There is no explicit indication of the direction of the parameter (input or output).</td>
<td>In WF 4, parameters are declared as arguments. Each argument has an explicit direction (in, out, or in/out).</td>
</tr>
<tr>
<td>Parameters</td>
<td>Parameters to an activity or workflow must be defined as public CLR or dependency properties. These properties must be defined in code.</td>
<td>Parameters to an activity or workflow are declaratively added to the model as arguments. Arguments can also be added to code-based activities.</td>
</tr>
<tr>
<td>Passing values</td>
<td>Values may be passed between activities by directly binding a property of one activity to the property of another.</td>
<td>Use variables to define intermediate storage instead of directly binding the properties of one activity to another.</td>
</tr>
<tr>
<td>Rule engine</td>
<td>A general purpose rules engine is included with WF 3.x. This rules engine can be executed directly in code or from the workflow model using the PolicyActivity.</td>
<td>WF 4 does not include a rules engine. However, the WF 3.x engine can be invoked directly in code from a custom WF 4 activity. The WF 3.x PolicyActivity can be invoked using the Interop activity.</td>
</tr>
<tr>
<td>Runtime</td>
<td>A single instance of the WorkflowRuntime class can be created and used to execute multiple workflow instances. The WorkflowServiceHost class is used to manage multiple instances of workflows that use WCF services.</td>
<td>A separate instance of the WorkflowApplication class is created for each workflow instance. The WorkflowServiceHost class is used to manage multiple instances of workflow services.</td>
</tr>
<tr>
<td>Services</td>
<td>Custom services may be developed and added to the workflow runtime. This makes their functionality available to the host application and workflow instances.</td>
<td>Custom workflow extensions can be developed and added to the two full-featured hosting environments (WorkflowApplication and WorkflowServiceHost).</td>
</tr>
</tbody>
</table>
Continued

<table>
<thead>
<tr>
<th>Category</th>
<th>WF 3.x</th>
<th>WF 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specialized base class</td>
<td>In WF 3.x, you can use one of the available base classes to develop custom activities in code. However, these base classes (for example SequenceActivity or CompositeActivity) all provide the same basic access to the workflow runtime. They each provide a varying feature set, but they are not designed to target a particular type of custom activity in the same way as the WF 4 base classes.</td>
<td>WF 4 provides multiple activity classes that can be used as the base class when you develop custom activities in code. Each base class is designed to meet the targeted needs of a particular type of custom activity and provides just enough access to the workflow runtime for that activity type. Example base classes are CodeActivity, AsyncCodeActivity, and NativeActivity.</td>
</tr>
<tr>
<td>State machine</td>
<td>WF 3.x supports a state machine workflow type that can be used to declare event-driven state machine workflows.</td>
<td>WF 4 does not support a state machine. However, the Flowchart activity provides the flexibility to declare many types of workflows that would previously require a state machine. True state machine functionality is promised by Microsoft after WF 4 is released.</td>
</tr>
<tr>
<td>Tracking</td>
<td>The default persistence mechanism for workflow tracking is a SQL Server database.</td>
<td>The default output medium for workflow tracking is Event Tracing for Windows (ETW). Persistence to a SQL Server database requires the development of a custom InstanceStore.</td>
</tr>
<tr>
<td>Variable scope</td>
<td>Variables that are defined for a workflow are always in scope. All member variables of the workflow class are global and accessible to all activities of the workflow.</td>
<td>Variables have scope and can be declared only for activities that support them. Defined variables are visible and accessible only to the activity that defined them and its children.</td>
</tr>
<tr>
<td>WCF contracts</td>
<td>WCF contracts can be defined as an interface in code and then referenced by WF 3.x activities. WF 3.x also supports workflow-first authoring of WCF operations.</td>
<td>WCF contracts are inferred from the properties of message-related activities within the workflow model. This is similar to the workflow-first WCF authoring style that is supported by WF 3.x. Contract-first authoring and reuse of existing WCF contract interfaces is not supported in WF 4.</td>
</tr>
</tbody>
</table>
## WF 3.x to WF 4 Activities

The following table compares the list of WF 3.x and WF 4 standard activities. In cases where there isn’t an exact match between WF 3.x and WF 4, I list the closest match that could be used to provide similar functionality or solve a similar problem.

<table>
<thead>
<tr>
<th>WF 3.x Activity</th>
<th>WF 4 Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddToCollection</td>
<td></td>
</tr>
<tr>
<td>Assign</td>
<td></td>
</tr>
<tr>
<td>CallExternalMethodActivity</td>
<td></td>
</tr>
<tr>
<td>CancellationHandlerActivity</td>
<td>CancellationScope</td>
</tr>
<tr>
<td></td>
<td>ClearCollection</td>
</tr>
<tr>
<td>CodeActivity</td>
<td></td>
</tr>
</tbody>
</table>
Continued

<table>
<thead>
<tr>
<th>WF 3.x Activity</th>
<th>WF 4 Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompensatableSequenceActivity</td>
<td>CompensableActivity</td>
</tr>
<tr>
<td>CompensatableTransactionScopeActivity</td>
<td>CompensableActivity</td>
</tr>
<tr>
<td>CompensateActivity</td>
<td>Compensate</td>
</tr>
<tr>
<td>CompensationHandlerActivity</td>
<td>CompensableActivity</td>
</tr>
<tr>
<td>ConditionedActivityGroup</td>
<td></td>
</tr>
<tr>
<td>Confirm</td>
<td></td>
</tr>
<tr>
<td>CorrelationScope</td>
<td></td>
</tr>
<tr>
<td>DelayActivity</td>
<td>Delay</td>
</tr>
<tr>
<td>DoWhile</td>
<td></td>
</tr>
<tr>
<td>EventDrivenActivity</td>
<td>PickBranch</td>
</tr>
<tr>
<td>EventHandlersActivity</td>
<td></td>
</tr>
<tr>
<td>EventHandlingScopeActivity</td>
<td>ExistsInCollection</td>
</tr>
<tr>
<td>FaultHandlerActivity</td>
<td></td>
</tr>
<tr>
<td>FaultHandlersActivity</td>
<td></td>
</tr>
<tr>
<td>Flowchart</td>
<td></td>
</tr>
<tr>
<td>FlowDecision</td>
<td></td>
</tr>
<tr>
<td>FlowSwitch</td>
<td></td>
</tr>
<tr>
<td>HandleExternalEventActivity</td>
<td></td>
</tr>
<tr>
<td>ElseIfActivity</td>
<td>If</td>
</tr>
<tr>
<td>ElseIfBranchActivity</td>
<td>If</td>
</tr>
<tr>
<td>WF 3.x Activity</td>
<td>WF 4 Activity</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>InitializeCorrelation</td>
<td></td>
</tr>
<tr>
<td>Interop</td>
<td></td>
</tr>
<tr>
<td>InvokeMethod</td>
<td></td>
</tr>
<tr>
<td>InvokeWebServiceActivity</td>
<td>Send, SendAndReceiveReply</td>
</tr>
<tr>
<td>InvokeWorkflowActivity</td>
<td></td>
</tr>
<tr>
<td>ListenActivity</td>
<td>Pick</td>
</tr>
<tr>
<td>ParallelActivity</td>
<td>Parallel</td>
</tr>
<tr>
<td></td>
<td>ParallelForEach</td>
</tr>
<tr>
<td></td>
<td>Persist</td>
</tr>
<tr>
<td>PolicyActivity</td>
<td></td>
</tr>
<tr>
<td>ReceiveActivity</td>
<td>Receive, ReceiveAndSendReply</td>
</tr>
<tr>
<td></td>
<td>RemoveFromCollection</td>
</tr>
<tr>
<td>ReplicatorActivity</td>
<td>ForEach</td>
</tr>
<tr>
<td></td>
<td>Rethrow</td>
</tr>
<tr>
<td>SendActivity</td>
<td>Send, SendAndReceiveReply</td>
</tr>
<tr>
<td>SequenceActivity</td>
<td>Sequence</td>
</tr>
<tr>
<td>SetStateActivity</td>
<td></td>
</tr>
<tr>
<td>StateActivity</td>
<td></td>
</tr>
<tr>
<td>StateFinalizationActivity</td>
<td></td>
</tr>
<tr>
<td>StateInitializationActivity</td>
<td></td>
</tr>
<tr>
<td>SuspendActivity</td>
<td></td>
</tr>
</tbody>
</table>
## APPENDIX B: COMPARING WF 3.x TO WF 4

### Continued

<table>
<thead>
<tr>
<th>WF 3.x Activity</th>
<th>WF 4 Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch</td>
<td>Switch</td>
</tr>
<tr>
<td>SynchronizationScopeActivity</td>
<td></td>
</tr>
<tr>
<td>TerminateActivity</td>
<td>TerminateWorkflow</td>
</tr>
<tr>
<td>ThrowActivity</td>
<td>Throw</td>
</tr>
<tr>
<td>TransactedReceiveScope</td>
<td>TransactedReceiveScope</td>
</tr>
<tr>
<td>TransactionScopeActivity</td>
<td>TransactionScope</td>
</tr>
<tr>
<td>TryCatch</td>
<td>TryCatch</td>
</tr>
<tr>
<td>WebServiceFaultActivity</td>
<td>SendReply</td>
</tr>
<tr>
<td>WebServiceInputActivity</td>
<td>Receive</td>
</tr>
<tr>
<td>WebServiceOutputActivity</td>
<td>SendReply</td>
</tr>
<tr>
<td>WhileActivity</td>
<td>While</td>
</tr>
</tbody>
</table>
Symbols
[] square brackets, 58

A
Abort method, 132, 138, 693
Aborted delegate, WorkflowApplication class and, 128
aborting activities, 693
aborting workflows, 849
AbortInstanceOnTransactionFailure property (TransactionScope activity), 540
Action delegate (C#), 127, 197
Action property (PickBranch activity), 105
activities, 2–44, 71–110, 849. See also workflows
aborting, 693
activity definition vs. activity instance, 55
activity properties and, 634–645
activity templates for, 683–685
adding to Visual Studio Toolbox, 301
base classes and, 23, 76
cancelling, 564, 693
categories of work and, 47, 72
child. See child activities
collection-related, 195–228
command-line calculator sample application illustrating. See command-line calculator (sample application)
composite, 74
creating, workflow designer and, 756
custom. See custom activities
data flow of, 99
declarative activity model and, 45
defined, 2
executing individually, 40
Hello Workflow sample application illustrating, 5–17
life cycle of, 55
metadata store for, 633, 640–642
parent. See parent activities
procedural, 163–194
standard. See standard activities
state and, 57
as synonymous with workflows, understanding, 46, 55
terminating, 693, 858
understanding, 71–78
using, 97
WF 3.x, executing from Interop activity, 801, 806–842 

workflow declaration and, 8–12, 17 

Activities assembly, 7 

Activity class, 16, 77, 849 

DynamicActivity class and, 142, 741–747 

shipping charges calculator sample application and, 89 

activity templates, 683–685, 767 

ActivityAction activity, 290–302 

activity templates and, 683 

custom designers and, 657–661 

scheduling, 734–740 

ActivityAction class, 195, 290, 849 

deleagtes and, 290 

properties of, 291 

supporting ActivityAction and, 657 

scheduling ActivityAction and, 734 

ActivityBuilder class, 756, 765, 849 

ActivityContext class, 56, 485, 849 

ActivityDefinitionId property (TrackingProfile class), 575 

ActivityDelegate class, 850 

ActivityDesigner class, 630, 850 

ActivityFunc class, 290, 850 

ActivityInfo class, 571 

ActivityInstance class, 56, 691, 850 

ActivityMetaProperties property (Interop activity), 806 

ActivityProperties property (Interop activity), 806 

ActivityScheduledQuery, 577 

ActivityScheduledRecord, 572, 577, 598 

ActivityState class, 572 

ActivityStateQuery, 576, 594 

ActivityStateRecord, 571, 576 

ActivityType property (Interop activity), 806 

ActivityValidationServices class, 661, 663, 680 

ActivityXamlServices class, 142–144 

AddBaseAddress method, 362 

AddExtension method (WorkflowApplication class), 129 

AddImplementationVariable method, 715 

addition case, Switch activity and, 30 

AdditionalConstraints property (ValidationSettings class), 663, 680 

addresses, endpoints and, 316 

AddServiceEndpoint method, 362 

AddToCollection activity, 101, 109, 198 

AddToDictionary activity, 214, 215 

AddValidationError method, 667 

AdventureWorks SQL Server sample database, 508–529 

downloading/installing, 509 

LINQ to SQL for accessing, 510 

tracking workflows and, 579 

Annotations property (TrackingRecord class), 568 

annotations, to tracking records, 568, 597 

anonymous functions, Lambda expressions and, 93
App.config file
  credit approval sample workflow service and, 400
order processing sample workflow service and, 344, 346, 359
order shipment sample workflow service and, 384, 388
self-hosting workflow services and, 366
shopping cart sample workflow and, 440, 455, 464
tracking profiles read from, 575, 623–626
WorkflowControlEndpoint and, 480
applications, 35–37
debugging, 37
external, guidelines for workflow interaction with, 266
quitting, 36
running. See running applications
workflow persistence and, 416
Arg prefix, 18, 26
ArgExpression argument, 25, 235
Argument Editor, 18, 25
argument properties, 21
ArgumentException, 21
arguments, 75, 850
  activity data flow and, 99
defining, 24, 25, 235
DesignerHost sample application, 796
dictionaries and, 118
properties and, 118
shipping charges calculator sample application and, 84
Arguments button (workflow designer), 10, 18
ArgumentToExpressionConverter class, 631, 637
arithmetic operations, Switch activity and, 29
ASP.NET web applications, invoking workflows from, 145–149
assemblies, 51
Assign activity, 29, 101, 107
  expressions and, 632
vs. InvokeMethod activity, 222
AsyncCodeActivity class, 47, 73, 76, 850
  inventory lookup sample and, 189
  shipping charges calculator sample application and, 94
  suspending persistence and, 419
AsyncCodeActivityContext class, 56
asynchronous unit of work, 73, 850
atomic unit of work, 73, 411, 850
Attributes property (ActivityBuilder class), 756
AutoResetEvent, 123, 125, 134, 273

B
base classes, 23, 76
generic/nongeneric versions of, 76
HostingDemoActivity sample and, 121
BasicHttpBinding, 340
BeginCancel method, WorkflowApplication class and, 132
BeginExecute method
inventory lookup sample and, 190
shipping charges calculator sample application and, 94
BeginInvoke method, 486
shipping charges calculator sample application and, 95
WorkflowInvoker class and, 113, 122, 124
BeginInvokeCallback method, 125
BeginLoad method (WorkflowApplication class), 130
BeginOnLoad method (PersistenceIOParticipant class), 471
BeginOnSave method (PersistenceIOParticipant class), 471
BeginPersist method (WorkflowApplication class), 130
BeginResumeBookmark method (WorkflowApplication class), 132
BeginRun method (WorkflowApplication class), 130, 140
BeginTerminate method (WorkflowApplication class), 132
BeginTrack method (TrackingParticipant class), 605
BeginTryCommand (InstanceStore class), 484, 486
BeginUnUnload method (WorkflowApplication class), 132

behaviors
workflow services and, 412
WorkflowServiceHost persistence and, 443

bindings, 851
endpoints and, 316
migration and, 805
properties and, 297
Web.config file and, 342
workflow services and, 316

Body property
CancellationToken activity, 108, 564
CompensableActivity, 108, 549
DoWhile activity, 104, 166
ForEach activity, 105, 195
ParallelForEach activity, 196
TransactionScope activity, 540
While activity, 104, 166

BookmarkCallback delegate, 267
BookmarkOptions enum, 692, 747
BookmarkResumptionQuery, 577
BookmarkResumptionRecord, 572, 577

bookmarks, 49, 266–276, 851
BookmarkResumptionQuery and, 577
BookmarkResumptionRecord and, 572
long-running unit of work and, 74
migration and, 805
nonblocking, 747, 748
parent/child activities and, 692
Pick/PickBranch activities and, 302–312
shopping cart sample workflow and, 429
workflow instances and, 131
Boolean conditions, evaluating/testing, 690, 702, 726
Branches property (Parallel activity), 105, 176
branching decisions
decision-making constructs for, 164
procedural modeling style and, 2, 3, 29
breakpoints, 37
building blocks, workflows and, 97

C

C#
expression support for, 68
vs. Visual Basic, 58–62
C# events, vs. delegates, 128
CacheMetadata method, 658, 851
configuring metadata and, 688
validation code and, 662, 667
CallExternalMethodActivity
ExternalDataExchangeService and, 823
Interop activity and, 807, 823
Cancel method, 463
child cancellations and, 693
WorkflowApplication class, 132, 137
CancelChild method
(NativeActivityContext class), 693, 728
CancelChild method
(NativeActivityFaultContext class), 692
CancelChildren method
(NativeActivityContext class), 693
CancelHandler property
(CancellationScope activity), 108
Canceled state, 57, 851
cancellation requests, 693
CancellationHandler property
CancellationScope activity, 564
CompensableActivity, 549
CancellationScope activity, 101, 108, 564
CancelRequestedQuery, 577
CancelRequestedRecord, 572, 577
CanInduceIdle property, 267, 269
case sensitivity, Visual Basic and, 59
cases, arithmetic operations and, 30
Cases property (Switch activity), 165
Catches property (TryCatch activity), 109, 530
ChannelFactory class, 317
child activities, 9
ActivityScheduledRecord and, 572, 598
cancelling, 693
CancelRequestedQuery and, 577
CancelRequestedRecord and, 572
exception handling for, 709–714
execution completion and, 691
execution properties and, 748
execution repeated for, 702–709
execution scheduling for, 690, 693–702,
714–725, 857
Parallel activity and, 176, 194
parallel execution for, 725–734
Sequence activity and, 176
single/multiple, 649–657, 667–683
Switch activity and, 85
ClearCollection activity, 101, 109, 198, 218
ClearDictionary activity, 214, 217
collection-related activities, 195–228
Close method (WorkflowServiceHost class), 362
Closed delegate, WorkflowApplication class and, 137
Closed event (WorkflowServiceHost class), 362
Closed state, 57, 851
Closing event (WorkflowServiceHost class), 362
CLR (Common Language Runtime). See runtime
CLR data types, variables and, 56, 75, 90
Code Activity template, 23
CodeActivity class, 23, 46, 64, 73, 851
access to the workflow runtime and, 76
migration and, 804
shipping charges calculator sample application and, 78
WF 3.x workflows and, 821
CodeActivityContext class, 24, 56
collapsed viewing mode, 645–649
collection activities, 198–211
collection category (of standard activities), 109
Collection property
AddToCollection activity, 109, 198
ClearCollection activity, 199
ExistsInCollection activity, 199
RemoveFromCollection activity, 198
collectionValues method
(PersistenceParticipant class), 470
command-line calculator (sample application)
ActivityAction used with, 293–302
alternate extension used with, 286–289
bookmarks used with, 268–276
flowchart modeling style and, 233–241
hosting workflows and, 34
ParseCalculatorArgs activity for, 23, 27, 234, 236
procedural modeling style and, 22–43
running, 35
unit testing for, 39–43
workflow extensions used with, 276–286
Common Language Runtime. See runtime; CLR data types, variables and
communication, 265–312
CommunicationObject class, 361
CompensableActivity, 101, 108, 549–557
Compensate activity, 101, 108, 550, 557
compensation, 50, 549–563, 852
CompensableActivity and, 549–557
compensation handlers and, 57
triggering manually, 557–563
compensation activities (WF 3.x), Interop activity and, 807
CompensationHandler property
(CompensableActivity), 549
Completed delegate, 127, 128, 273
CompletionCondition property
Parallel activity, 105, 176, 188, 194
ParallelForEach activity, 196
CompletionCallback callback delegate, 691
CompletionState property, 134
complex types, controlling serialization of, 323
composite activities, 9, 74, 852
composition, 852
Condition property, 106
Boolean conditions and, 702
child activity scheduling and, 714
DoWhile activity, 104, 166
If activity, 164
While activity, 104, 166
ConfigurationManager class, 623
configuring workflow services, 316
Confirm activity, 101, 108, 550
confirmation, 550, 852
ConfirmationHandler property
(CompensableActivity), 549
consistency of database updates
compensation and, 549–557
TransactionScope activity for, 541–546, 549
Console class, 13
Constraint class, 662
Constraints property (ActivityBuilder class), 663, 756
constraints, for validation, 662, 669–679, 852
Content property, messaging activities
and, 323, 324
content-based correlation, 325, 386–388, 852
ContentControl class, 630
ContentPresenter class, 648
correlation, 325, 370, 852
controlling, 325
workflow services and, 324
CorrelationHandle, 325
CorrelationInitializer class, 325
CorrelationScope activity, 101, 107, 326
CreateBookmark method
(NativeActivityFaultContext class), 267, 692
CreateInstanceStore method, 436
CreateWorkflowOwnerCommand, 421, 422, 444, 485
credit approval (sample) workflow service, 388–405
declaring, 390–393
order processing sample workflow service modified for, 394–399
testing, 402–405

custom activities, 23, 46, 71, 687–752
adding to Sequence activity, 28
asynchronous implementation for, 93–97
authoring, ways of steps for, 72, 78, 97
cancelling, 564
code authoring for, 78–84, 88–93
custom tracking records and, 600–604
CustomTrackingRecord for, 573
designers for, 47, 629–634
dictionary-related activities and, 213–222
executing rules via, 842–847
expression activities and, 64–66
flowchart modeling style and, 248–253
LINQ queries/updates and, 511–520
migration and, 804
order processing sample workflow service and, 349
parents/children and, 687–725
for shopping cart sample workflow, 428
tracking workflows and, 580
unit testing for, 39
validation for, 661–683
workflow building blocks and, 97
workflow extensions and, 406
XAML authoring for, 84–88
CustomTrackingQuery, 578
CustomTrackingRecord, 573, 578

D

data
categories of, activities and, 74
data contracts and, 315, 322, 330, 852
workflow persistence and, 419
data flow of activities, 99
data types, C# vs. Visual Basic, 61
DataContract attribute, 315, 322, 331, 419
DataMember attribute, 315, 322, 331, 419
debugging applications, 37, 48
decisions, 2, 22–38, 164, 233–241
declarative activity model, 45, 853
declarative workflow model, 2
declarative workflow services, tracking workflows and, 618–623
Default case, Switch activity and, 30, 32, 36
Default property (Switch activity), 165
definition, 55
Delay activity, 101, 107, 114
adding to a workflow, 180
Parallel activity and, 177
SqlWorkflowInstanceStore persistence and, 422
TimeSpan value and, 120
workflow persistence and, 418
delegates
  vs. C# events, 128
  multiple instances of workflows and, 152
  WorkflowApplication class and, 127
DeleteWorkflowOwnerCommand, 421, 422, 485
shopping cart sample workflow and, 437
WorkflowServiceHost and, 444
Designer attribute, 633, 639
  commenting out for testing purposes, 642
  using vs. metadata store, 642
DesignerHost (sample) application, 760–799
  context menu, modifying for, 793–796
  controls for, 761
  currently selected activity, tracking, 790–793
  implementing, 760–771
Save/SaveAs/Open operations and, 775–777
steps for implementing (list), 760
Toolbox for, 761–771, 780–788
DesignerMetadata class, 760, 764, 788
designers, 629–634
  associating with activities, 633, 639–642
  default, 636
  expanded/collapsible, 645–649
shipping charges calculator sample application illustrating, 635–649
steps for creating (list), 634
design-time
  custom activities and, 629–685
design-time audience and, 72
developers, activities and, 72
dictionaries
  arguments and, 112, 118
dictionary-related activities for, 213–222
DisplayName property, 12
division case, Switch activity and, 31
domain-specific languages, workflow model and, 2
DoWhile activity, 101, 104, 165, 174, 632
downloads
  AdventureWorks SQL Server sample database, 509
  sample code for Pro WF (in .NET 3.5) book, 824
  sample code for this book (Pro WF in .NET 4.0), 761
duplex message exchange pattern, 315, 388–405, 853
correlation and, 324
guidelines for, 389
durable delay, 853
DurableInstancingOptions property, 362
DynamicActivity class, 77, 142, 741–747, 853
DynamicActivityProperty class, 743
e-commerce shopping cart (sample) workflow. See shopping cart (sample) workflow
Else property (If activity), 104, 164
EditingContext class, 757, 853
EndExecute method
  inventory lookup sample and, 191
  shipping charges calculator sample application and, 94, 96
EndInvoke method, 486
  inventory lookup sample and, 191
  shipping charges calculator sample application and, 96
  WorkflowInvoker class, 113, 122, 124
EndOnLoad method
  (PersistenceIOParticipant class), 471
EndOnSave method
  (PersistenceIOParticipant class), 471
endpoints, 316, 479–483, 853
EndRun method (WorkflowApplication class), 130, 141
EndTrack method (TrackingParticipant class), 605
EndTryCommand (InstanceStore class), 484, 486
Enqueue method (WF 3.x), migration and, 805
EnqueueOnIdle (WF 3.x), migration and, 805
error handling category (of standard activities), 109
ETW (Event Tracing for Windows), 68, 565, 578, 853
ETW tracking participant, 579–591
EtwTrackingParticipant class, 277, 578
event logging, 578
Event Tracing for Windows (ETW), 68, 565, 578, 853
event tracking, EtwTrackingParticipant for, 277, 578
EventData element, tracking data and, 589
events
  custom, migration and, 805
  vs. queues, 286
EventTime property (TrackingRecord class), 568
examples. See samples
Executing state, 57
exception handling, 50, 57, 507–530
  ArgumentException and, 21
  FaultPropagationQuery for, 577
  FaultPropagationRecord for, 573
  multiple exceptions and, capturing, 537
  parents/children and, 692, 709–714
  Throw activity and, 32
  unhandled exceptions and, 507
  workflow services and, 409
Execute method, 266, 268
  command-line calculator sample application and, 24
  inventory lookup sample and, 168
  shipping charges calculator sample application and, 79, 94, 810
Executing state, 853
execution properties, 747–751
ExistsInCollection activity, 101, 109, 198, 201
ExistsInDictionary activity, 214, 216
expanded viewing mode, 645–649
expression activities, 62–66
Expression argument, 24, 28
Expression property
  FlowSwitch activity, 106
  Switch activity, 90, 104, 165
expressions, 50, 57–66, 854
designer classes and, 631
types of, 57, 632
Visual Basic for, 57–62, 859
Expressions namespace, 63
ExpressionTextBox class, 631, 637, 651
Extensible Application Markup Language. See Xaml
Extensions property (WorkflowApplication class), 129
ExternalDataExchange attribute, 823
ExternalDataExchangeService, WF 3.x and, 823–834, 854

file system
  file-based instance stores and, 485–492, 501
  workflow persistence and, 417, 484
FileSystemInstanceStore (sample) class, 485, 492
FileSystemInstanceStoreIO (sample) class, 485, 492–501
Finally property (TryCatch activity), 109, 530, 537
FindInCollection activity, 199, 201, 255
FindInDictionary activity, 214, 217
FirstNumber argument, 24–28, 30
flowchart activities, 230–263
Flowchart activity, 9, 101, 105, 230, 854
  command-line calculator sample application and, 233
  context menu and, 793–796
  Toolbox items and, 790–793
flowchart category (of standard activities), 105
flowchart modeling style, 3, 47, 163, 229–263, 854
  mixing with procedural modeling style, 229, 254–263
  using 230–233
FlowchartDecision activity, 230, 231
FlowchartStep activity, 230, 232
FlowchartSwitch activity, 230, 231
FlowDecision activity, 102, 106, 854
  context menu and, 793–796
  Toolbox items and, 790–793
FlowNode class, 230

fault
  callback callback delegate, 691
  Faulted event (WorkflowServiceHost class), 362
  Faulted state, 57, 854
  FaultPropagationQuery, 577
  FaultPropagationRecord, 573, 577
FlowSwitch activity, 102, 106, 854
  context menu and, 793–796
  Toolbox items and, 790–793
Flush method (WorkflowDesigner class), 754, 772
ForEach activity, 102, 105, 195, 200–211, 254
  activity templates and, 683
  ActivityAction activity and, 734
  ActivityAction class and, 197, 657
foreach keyword (C#), 195
ForEachWithBodyFactory class, as activity template, 683, 767
Func delegate, 89, 95, 127, 486

G
GenericTypeArguments property
  (InvokeMethod activity), 223
generic types, C# vs. Visual Basic, 59
Get method, 24, 80
GetAdditionalExtensions method
  (IWorkflowInstanceExtensionExtension
  interface), 278
GetBookmarks method
  (WorkflowApplication class), 132
GetChildSubtree class, 672
GetExtension method
  (WorkflowApplication class), 129, 277
GetExtensions method
  (WorkflowApplication class), 129
GetParentChain class, 672, 675
GetWorkflowTree class, 672

globally unique identifiers GUIDs),
  workflow instances and, 137
glossary, 849–860
guessing game (sample) application, 824–834
GUIDs (globally unique identifiers),
  workflow instances and, 137

H
HandleExternalEventActivity
  ExternalDataExchangeService and, 823
  Interop activity and, 807, 823, 807, 823
HandleFault method
  (NativeActivityFaultContext class), 692, 712
Hello Workflow (sample application), 5–17
  adding new workflows to, 17
  hosting workflows and, 12, 20
  running, 14
host communication, 265–312
hosting workflow services, 49
  credit approval sample workflow
  service and, 400
  order processing sample workflow
  service and, 359
  order shipment sample workflow
  service and, 384
  self-hosting and, 361–368
  Xaml workflows and, 326
  Xamlx workflow services and, 326
hosting workflows, 48
  command-line calculator sample
  application and, 34, 239
Hello Workflow sample application and, 12, 20
HostingDemoActivity sample illustrating, 121
HostingDemoWorkflow sample illustrating, 117
WebInvokerHost and, 147
WorkflowApplication class and, 126–142
WorkflowInvoker class and, 111–126
Xaml workflows and, 326
HostingDemoActivity (sample), 121
HostingDemoWorkflow (sample)
ActivityXamlServices class and, 143
hosting the workflow, 117
multiple workflow instances and, 150
WorkflowApplication class and, 133
WorkflowInvoker class and, 114–120
workflows invoked from ASP.NET web applications and, 145
human interactions, 855
workflow model and, 2
workflow persistence and, 416

I
IActivityTemplateFactory interface, 683
IAsyncResult object, 124, 125
icons, 643
Idle delegate, 127, 128, 137, 273
Idle member, PersistableIdle member and, 420
IEquatable interface, 201, 202
If activity, 102, 104, 164
problem reporting sample workflow and, 303
shipping charges calculator sample application and, 85, 90
if keyword, 48
if statement (C#), 85
IIS (Internet Information Services), 316
publishing workflow services to, 341
workflow persistence and, 416
images, 643
imperative code, migration and, 804
Implementation property
Activity class, 89
ActivityBuilder class, 756
Imports button (workflow designer), 10
InArgument class, 21, 24, 75, 90
inferred contracts, 324
infrastructure, work categories and, 74
initialization syntax, vs. procedural code, 742
Initialize method/Uninitialize method (WF 3.x), migration and, 805
InitializeCorrelation activity, 102, 107
InitializeDesigner method, 764, 779
input arguments, 18, 75
input parameters, 17–22
instance methods, WorkflowInvoker class and, 112, 122–126
instance stores, 416, 483–505, 855
InstanceId property (TrackingRecord class), 568
InstanceInfo class (C#), 154, 156
InstancePersistence class, 484
InstancePersistenceCommand class, 484
InstancePersistenceContext class, 485
InstanceStore class, 416, 484, 485, 855
InstanceStore property, 130, 420, 443
InstanceView property
(InstancePersistenceContext class), 485
IntelliSense, 19
Internet Information Services (IIS), 316
publishing workflow services to, 341
workflow persistence and, 416
Interop activity, 51, 102, 110, 806–842, 855
ExternalDataExchangeService and,
823–834
limitations of, 807
migration and, 804
rules, executing via, 834–842
shipping charges calculator sample
application illustrating, 807–823,
834–842
WF 3.x activities, executing from, 801,
806–842
Interop class, 806
InteropTrackingRecord class, 574, 807
inventory lookup (sample), 241
hosting/testing, 172
Parallel activity and, 182–194
While/DoWhile activities and, 167–175
inventory update (sample), 200–213, 254–
263, 508–564
custom activities for, 511–530
dictionary-related activities and, 218–
222
executing, 526
ForEach activity and, 200–211
InvokeMethod activity and, 223–227
ParallelForEach activity and, 211
tracking workflows and, 579–605, 611–
626
Invoke method, 13, 35, 111
InvokeAction activity, 290, 300
InvokeAction class, 290
InvokeAsync method (WorkflowInvoker
class), 113, 122
InvokeCompleted event, 123
InvokeFunc class, 290
InvokeMethod activity, 102, 108, 222–227
InvokeWorkflowActivity, Interop activity
and, 807
IsCancellationRequested property, 704
IsolationLevel property (TransactionScope
activity), 540
Item property
addToCollection activity, 109, 198
RemoveFromCollection activity, 198
Items property
EditingContext class, 757
ExistsInCollection activity, 199
ItemSupportExtension (sample) custom
workflow extension, 426, 472–474
IValidationErrorResponseService interface, 777
IWorkflowInstanceExtension interface, 278
keywords, C# vs. Visual Basic, 59

Lambda expressions, 90, 93, 855
Language INtegrated Query (LINQ), 93, 855, 510
Level property (TrackingRecord class), 568
LINQ (Language INtegrated Query), 93, 855, 510
LINQ to SQL, 510, 855
Literal class, 58
literal data types, C# vs. Visual Basic, 62
Load method
  resuming persistence after suspending, 420
  WorkflowApplication class and, 130, 417
  WorkflowDesigner class and, 754, 755
  Xaml and, 142
LoadWorkflowByKeyCommand, 485
LoadWorkflowCommand, 485
local services, 823
long-running unit of work, 73, 855
long-running workflows, 265, 327
  bookmarks for, 266–276
  cancelling, 564
  WorkflowControlEndpoint for, 479–483
looping decisions, 165
  procedural modeling style and, 2, 3, 34
  flowchart modeling style and, 241–248
L-value expressions, 632, 639

management endpoints, 479–483
MapValues method
  (PersistenceParticipant class), 470
message-based communication. See workflow services
message contracts, 323, 330, 856
message exchange patterns, 315, 317, 856
MessageBodyMember attribute, 323, 324, 331
MessageContract attribute, 323, 324, 331
MessageHeader attribute, 323, 324, 331
MessageReceived event,
  ExternalDataExchangeService and, 831
messaging activities, 317–322, 324
messaging category (of standard activities), 106
metadata
  configuring automatically/manually, 688–690
  custom instance stores and, 490–498
  DesignerMetadata class and, 760, 764, 788
  MetadataStore class and, 633, 760, 856
metadata store, 633, 640–642
MetadataStore class, 633, 760, 856
MethodName property (InvokeMethod activity), 223
migration category (of standard activities), 110
migration strategies, 51, 801–806
ModelChanged event (WorkflowDesigner class), 754, 764
modeling styles. See flowchart modeling style; procedural modeling style
Modelltem class, 630, 856
Modelltem property (ActivityDesigner class), 630, 638
multiplication case, Switch activity and, 31

N
Name property
   ActivityBuilder class, 756
   TrackingProfile class, 575
namespaces, 7, 10, 51
naming conventions
   for assemblies, Visual Studio and, 640
   for service contracts, 319
NativeActivity, 856
NativeActivity class, 74, 266, 291
   access to the workflow runtime and, 76
   configuring metadata and, 688
   executing activities and, 694
NativeActivityContext class, 56, 267, 291
   cancellation requests and, 693
   child activity execution scheduling and, 690
   execution properties and, 747
   suspending persistence and, 419
NativeActivityFaultContext class, 692
NativeActivityMetadata struct, 689, 735
.NET Framework
   examples in this book and, 6
   projects and, 6
   WF and, 4, 803
nonblocking bookmarks, 747, 748
nonflowchart activities, 232
nonpersisting tracking participants, 615–618
NonSerialized attribute, workflow persistence and, 419
nonworkflow applications, implementing, 342–348
NoPersistHandle class, suspending persistence via, 419
notification to external systems, 509, 517, 551

O
object initialization, C# vs. Visual Basic, 59
Of keyword, 59
OnComplete method, 695, 704, 727
OnConditionComplete method, 703, 715
one-way message exchange pattern, 315, 856
OnFaulted method, 704, 712
OnUnhandledException delegate, 128, 135, 273
OnUnhandledException method, 507, 527, 529
Open method (WorkflowServiceHost class), 362
Opened event (WorkflowServiceHost class), 362
Opening event (WorkflowServiceHost class), 362
Operation argument, 24, 28
OperationContract attribute, 324
operators, C# vs. Visual Basic, 59
order processing (sample) workflow service, 313, 328–368
classes for, 329–332
client application for, 342–348
configuring/testing, 339
credit approval sample workflow service and, 388–405
declaring, 332–336
execution properties and, 747–751
order shipment sample workflow service and, 369–388
response for, populating, 337
workflow client for, 348–361
workflow extensions and, 405–409
order shipment (sample) workflow service, 369–388
declaring, 371–379
order processing sample workflow service modified for, 379–383
testing, 385, 388
OrderEntry (sample) workflow
configuring/testing and, 440
declaring, 431
OrderEntryService (sample) workflow service
client application and, 458
configuring/testing, 464, 477
declaring, 445
promoting properties and, 476–479
OrderUtilityExtension (sample), 406–409
order processing sample workflow service modified for, 407
testing, 409
orphaned workflow instances, 443
OutArgument class, 21, 24, 75
output arguments, 18, 75
OverloadGroup validation attribute, 662, 664

P
Parallel activity, 102, 105, 176–188
asynchronous execution via, 189
child activities and, 176, 194
parallel execution, 176, 189, 725–734, 856
ParallelForEach activity, 102, 105, 211
ActivityAction class and, 197
vs. ForEach activity, 196
ParallelForEachWithBodyFactory activity, 767
parameters, passing to workflows, 17–22
Parameters property (InvokeMethod activity), 223
parent activities
cancelling, 693
constrained activities and, 674
exception handling for, 709–714
execution properties and, 748
responsibilities of, 687–693
validation and, 678
ParseCalculatorArgs (sample) activity
  implementing in command-line calculator sample application, 234, 236, 269
  unit testing for, 39
performance
  argument properties and, 22
  collection-related activities and, 199
Persist activity, 102, 107, 417
Persist method (WorkflowApplication class), 130, 417, 420
PersistableIdle delegate,
  WorkflowApplication class and, 128, 136
PersistableIdle member, 420, 438, 442
PersistableIdle property,
  WorkflowServiceHost and, 443
persistence, 3, 49, 415–467, 857
  arguments/variables and, 75
  configuring/managing, 129
  customizing, 469–505
  defined, 415
  execution properties and, 747
  file system and, 417, 484
  how it works, 417
  migration and, 807
  nonpersisting tracking participants and, 615–618
  persistence commands (list), 484
  reasons for using, 415
  resuming after suspending, 420
SqlWorkflowInstanceStore and, 421–467
  suspending, 419
  tracking participants and, 604–614
WorkflowApplication and, 419, 424–443
WorkflowServiceHost and, 443–467
PersistenceIOParticipant class, 471
PersistenceParticipant class, 419, 469, 472–474, 857
Pick activity, 102, 105, 302–312, 371, 378
PickBranch activity, 102, 105, 371, 378
PolicyActivity class, 834, 842
Presentation namespace, 753
primitives category (of standard activities), 107
problem reporting (sample) workflow, 303–312
procedural activities, 163–194, 230
procedural code, vs. initialization syntax, 742
procedural model, 1, 45
procedural modeling style, 3, 47, 163, 857
  branching decisions and, 2, 3, 29
  mixing with flowchart modeling style, 229, 254–263
Program.cs file, 7, 12, 20
projects
  creating, 6
  running, 15
Promote method
(SqlWorkflowInstanceStore class)
promoting properties, 474–479
properties
activity properties and, 634–645
binding, 297
execution properties and, 747–751
promoting, 474–479
Properties property (ActivityBuilder class), 756
Properties window, 11
PropertyInspectorView property
(WorkflowDesigner class), 754, 755
PropertyReference activity, 297
proxy classes, client applications and, 316, 344
PublishValues method
(PersistenceParticipant class), 470

Q
Queries property (TrackingProfile class), 575
Query method, 463
Queue class, 286
queues, 286
quitting applications, 36

R
Receive activity, 102, 106
credit approval sample workflow service and, 394
inferred contracts and, 324
Interop activity and, 807
SendReply activity and, 319
WCF and, 317
ReceiveAndSendReply activity, 102, 106, 683
ReceiveAndSendReply activity template, 319
ReceiveMessageRecord, 574
ReceiveReply activity, 317, 321
RecordNumber property (TrackingRecord class), 568
referencedCategories dictionary, 781
reflection, activity metadata and, 690
Register method (DesignerMetadata class), 760, 788
RemoveFromCollection activity, 102, 109, 198
RemoveFromDictionary activity, 214, 215
RenderingInfo element, tracking data and, 589
request/response message exchange pattern, 315, 857
correlation and, 325
order processing sample workflow service and, 327–368
RequiredArgument validation attribute, 662, 663
RequiredArgumentAttribute, 24
resources, workflow persistence and, 416
resources for further reading
Lambda expressions, 93
LINQ, 511
WCF, 314
WF 3.x, 811
WF 3.5, 629
Result argument, 25
Result property
CompensableActivity, 549, 557
ExistsInCollection activity, 199
InvokeMethod activity, 223
RemoveFromCollection activity, 198
ResumeBookmark method, 267
resuming persistence after suspending, 420
WorkflowApplication class and, 132
Rethrow activity, 102, 109
root (topmost) activity, 9
rules engine, 68, 857
custom activities and, 842–847
Interop activity and, 834–842
RuleSet class, 836, 842, 857
Run method
resuming persistence after suspending, 420
WorkflowApplication class and, 130, 151
RunAsynchronously property
(InvokeMethod activity), 223
running applications
command-line calculator sample application and, 35
Hello Workflow sample application and, 14, 21
runtime, 851
activity definition vs. activity instance, 55
ActivityAction class and, 290
DynamicActivity class and, 741
private variables and, 56
runtime category (of standard activities), 107
samples
command-line calculator. See command-line calculator (sample application)
credit approval workflow service, 388–405
DesignerHost application, 760–799
FileSystemInstanceStore class, 485
FileSystemInstanceStoreIO class, 485, 492–501
guessing game application, 824–834
Hello Workflow, 5–17
HostingDemoWorkflow. See HostingDemoWorkflow (sample)
inventory lookup, 167–175, 182–194, 241
inventory update. See inventory update (sample)
ItemSupportExtension custom workflow extension, 426, 472–474
order processing workflow service. See order processing (sample)
workflow service
order shipment workflow service, 369–388
OrderEntry workflow, 431, 440
OrderEntryService workflow service, 445, 458, 464
OrderUtilityExtension, 406–409
problem reporting workflow, 303–312
sample code for this book, downloading, 761
shipping charges calculator. See shipping charges calculator (sample application)
shopping cart workflow, 424–467
Save method (WorkflowDesigner class), 754
SaveWorkflowCommand, 485
scalability, workflow persistence and, 416
ScheduleAction method
(NativeActivityContext class), 291, 690, 734
ScheduleActivity method
(NativeActivityContext class), 690, 695, 727
scheduledChildren dictionary, 726
ScheduleDelegate method
(NativeActivityContext class), 690, 734
ScheduleFunc method
(NativeActivityContext class), 291, 690, 734
scope, variables and, 26, 100
SecondNumber argument, 24–28, 30
self-hosting workflow services
steps for implementing, 361–368
testing/using, 367
WorkflowApplication class and, 416
WorkflowServiceHost class and, 416, 422
Send activity, 102, 106, 317, 320
Interop activity and, 807
ReceiveReply activity and, 321
SendAndReceiveReply activity, 102, 106, 683
SendAndReceiveReply activity template, 322
SendMessageRecord, 574
SendReply activity, 317, 319, 324
Sequence activity, 9, 102, 104
adding activities to, 9, 28
child activities and, 176
command-line calculator sample application and, 233
custom designers and, 653
Parallel activity and, 179
shipping charges calculator sample application and, 90
testing constraints and, 677
serializing tracking records to XML, 605–608
service contracts, 315
endpoints and, 316
naming conventions for, 319
service operations, order processing
sample workflow service and, 327–368
ServiceContract attribute, 324
ServiceHostBase class, 361
ServiceModel Metadata Utility (svcutil.exe utility), 316
Services property (EditingContext class), 757

Set method, 24, 80

SetChildrenCollection method, 715

SetInstance method
(IWorkflowInstanceExtension interface), 278

shipping charges calculator (sample application), 78–97

custom activity asynchronous implementation for, 93–97

custom activity code authoring for, 78–84, 88–93

custom activity designers and, 635–649

custom activity Xaml authoring for, 84–88

DynamicActivity and, 741–747

flowchart modeling style and, 248–253

project for, 79

unit testing for, 81–84, 87, 92, 96, 251

validation and, 663

WF 3.x activities and, 807–823

WF 3.x workflows and, 815–823, 834–842

shopping cart (sample) workflow, 424–467

configuring/testing, 440, 464

item-related classes for, 425

tasks for implementing (list), 424, 444

workflow implementation for, 424–443

workflow service implementation for, 444–467

ShowValidationErrors method, 777

SOAP fault handling, workflow services and, 409

SqlWorkflowInstanceStore, 421–467

Delay activity and, 422

properties of, 421

SQL Server preparation for, 423

using with WorkflowApplication, 424–443

using with WorkflowServiceHost, 444–467

SqlWorkflowInstanceStore class, 53, 416, 443

property promotion and, 474–479

workflow persistence and, 420

SqlWorkflowInstanceStoreBehavior class, 454, 456

Delay activity and, 423

workflow persistence and, 422

WorkflowServiceHost persistence and, 443

square brackets [], serializing expressions and, 58

standard activities, 3, 46, 71, 101–110

categories of, 103–110

Interop activity and, 807

list of, 101

WF 3.x vs. WF 4 (table), 865–868

workflow building blocks and, 97

standard flow activities, 9

state

activities and, 57

ActivityState class and, 572
ActivityStateQuery and, 576
ActivityStateRecord and, 571
tracking profiles and, 592–596
tracking workflows and, 569
state machine, 67, 857
static methods, WorkflowInvoker class
and, 111, 114–122
String class, 19
string literals, 11
subtraction case, Switch activity and, 31
SuspendActivity, Interop activity and, 807
svcconfigeditor.exe (WCF Service
Configuration Editor), 342
svccutil.exe utility (ServiceModel Metadata
Utility), 316
Switch activity, 29, 102, 104, 164
FlowchartSwitch activity and, 231
shipping charges calculator sample
application and, 85, 90
switch statement (C#), 29, 104, 165, 231
SynchronizationContext class,
InvokeAsync method and, 113
SynchronizationContext property
(WorkflowApplication class), 130,
160
System element, tracking data and, 589
system interactions, 2, 858
system-provided bindings, WCF and, 316
TargetType property (InvokeMethod
activity), 223
templates, 6
Terminate method (WorkflowApplication
class), 132, 139
TerminateWorkflow activity, 102, 107
terminating activities, 693, 858
text, WriteLine activity for, 10
Text property
   WriteLine activity, 11, 12
   WorkflowDesigner class, 754, 755
TextWriter property
   HostingDemoWorkflow sample and,
   114
   WriteLine activity and, 108
Then property (If activity), 104, 164
Throw activity, 32, 103, 109
Timeout property (TransactionScope
activity), 540
timeout values, 119
TimeSpan value, 119, 131
TimeToPersist property
(WorkflowIdleBehavior class), 444
TimeToUnload property
(WorkflowIdleBehavior class), 444, 454
ToolboxCategory class, 755
ToolboxControl class, 755
ToolboxItemWrapper class, 755, 768
topmost (root) activity, 9
Track method (TrackingParticipant class),
578, 605, 609
Tracking namespace, 600

T
Target property (Compensate activity), 550
TargetObject property (InvokeMethod
activity), 223
tracking participants, 566, 578, 604–618, 858
  custom, 604–618
  tracking profiles and, 574
tracking profiles, 566, 574–578, 591–604, 858
  loading from App.config file, 623–626
  targeting selected activities and, 596
tracking records, 566, 568–574, 858
  custom, 600–604
  serializing to XML, 605–608
  tracking participants and, 578
  tracking profiles for, 574, 591
  viewing tracking data and, 587, 623
tracking workflows, 50, 565–627, 858
  architecture of the workflow tracking mechanism, 566
  declarative workflow services and, 618–623
  migration and, 807
  reasons for using, 566
  SQL Server and, 68
  steps for implementing (list), 567
  viewing tracking data and, 579, 586, 623
TrackingParticipant class, 578, 605
TrackingProfile class, 574, 575
TrackingProvider class, 566
TrackingRecord class, 568
TransactedReceiveScope activity, 103, 107, 412, 541
transaction and compensation category (of standard activities), 108
transactions, 50, 540–549, 858
  host transactions and, 547
  workflows/workflow services and, 411
TransactionScope class, 540
transport protocol, identified by binding, 316
Trigger property (PickBranch activity), 105
Try property (TryCatch activity), 109, 530
  try/catch blocks, 35, 531, 549
TryCatch activity, 103, 109, 410, 530–540
  multiple exceptions and, capturing, 537
  properties of, 530
TryCommand (InstanceStore class), 484, 488

U
Uniform Resource Identifier (URI),
  endpoint addresses and, 316
unit testing, 39–43
Unload method (WorkflowApplication class), 132
Unloaded delegate, WorkflowApplication class and, 128, 137
URI (Uniform Resource Identifier),
  endpoint addresses and, 316
using statement (C#), 10
V

Validate method
(ActivityValidationServices class), 661, 663, 680

validation, 47, 661–683, 858
  constraints for, 662, 669–679
  executing manually, 679–683
  mechanisms for (list), 662
  validation attributes for, 662, 663–666
  validation code for, 662, 667–669
  validation errors, displaying, 777
ValidationContext property, 672
ValidationSettings class, 663, 680

value expressions, 632

Values property
  ForEach activity, 105, 195
  ParallelForEach activity, 196

Variable class, 56, 75, 90

Variable Editor, 27

variables, 75, 858
  activity data flow and, 99
  defining, 26, 235
  scope and, 26, 100, 859
  shipping charges calculator sample application and, 85
  shopping cart sample workflow and, 431, 445

Variables button (workflow designer), 10, 26

Variables Editor, 26

VB (Visual Basic), for expressions, 57–62, 859

versioning, 803

View property (WorkflowDesigner class), 754, 755

viewing modes, expanded/collapsed, 645–649

visibility, 565

Visual Basic (VB), for expressions, 57–62, 859

Visual Studio
  .NET Framework and, 804
  privileges and, 329, 367
  unit testing and, 39
  WF 3.x and, 802
  WF and, 4
  workflow designer and. See workflow designer

Visual Studio debugger, 48

Visual Studio Toolbox, 6, 301

VisualBasicReference class, 58

VisualBasicValue class, 58

W

warnings, validation and, 669

WAS (Windows Process Activation Service), 316

WCA.exe utility, 823, 826

WCF (Windows Communication Foundation), 48, 314–317, 859
  message exchange patterns and, 315, 317
  WCF contracts and, 67
  WF integration with, 3
workflow services and. See workflow services

WCF Service Configuration Editor (svcconfigeditor.exe), 342

WCF Test Client (WcfTestClient.exe), 342

web service activities, Interop activity and, 807

Web.config file
  declarative workflow services and, 620
  enhancing for workflow services publishing to IIS, 341

WF (Windows Workflow Foundation), 859
  base classes and, 23
  creating workflow in, 15
  features missing from version 4.0 of, 66
  features/capabilities of, 45–51
  grand tour of, 45–69
  migration strategies for, 51, 801–806
  quick tour of, 1–44
  reasons for using, 3
  variable scope and, 27
  version 4.0 of, 4, 859
  WF 3.x and, 801–847, 859
  WF 3.x vs. WF 4 (table), 861–868

While activity, 103, 104, 165–175
  custom designers and, 650
  WorkflowItemPresenter/WorkflowItem Presenter classes and, 632

while keyword, 48

Windows Communication Foundation. See WCF

Windows Event Viewer, 579, 586, 623

Windows Presentation Foundation (WPF) applications, 131, 160, 629

Windows Process Activation Service (WAS), 316

Windows Server AppFabric, 316

Windows Workflow Foundation. See WF
  work categories, activities and, 72
  workflow clients, implementing, 348–361

Workflow Console Application template, 6, 23, 235

workflow designer, 4, 8, 47
  architecture/components of, 753–760
  DesignerHost sample application illustrating, 760–799
  hosting, 753–799
  interface of, 10
  rehosting, 51
  Visual Studio and, 47

workflow extensions, 49, 276–286, 859
  adding to workflows, 409
  alternate extension and, 286–289
  custom, 405–409, 426, 472–474
  implementing, 406
  workflow persistence and, 419

workflow projects
  creating, 6
  running, 15

workflow services, 48, 313–414, 860
  behaviors and, 412
  configuring, 316
  controlling instances of, 327
correlation and, 324
credit approval sample workflow service and, 388–405
declarative, 618–623
declaring, 327–340
exceptions/faults and, 409
hosting, 316
message types and, 322
messaging activities for implementing, 317–322
order processing sample workflow service. See order processing (sample) workflow service
order shipment sample workflow service. See order shipment (sample) workflow service
OrderEntryService sample workflow service and, 445, 458, 464
publishing to IIS, 341
self-hosting, 361–368
transactions and, 411
WCF contracts and, 67
workflow instances, identifying, 325
workflow tracking. See tracking workflows
workflow workflow, 4
Workflow1.xaml file, 7, 15–17
WorkflowApplication class, 49, 57, 126–142, 860
creating instances of, 127
Delay activity and, 418
delegate members and, 127
GetExtension method and, 277
handling unhandled exceptions and, 507
ResumeBookmark method, 267
SqlWorkflowInstanceStore used with, 424–443
workflow extensions and, 277
workflow instance execution and, 130
workflow persistence and, 416, 419
vs. WorkflowInvoker class, 126, 273
WorkflowControlClient class, 480
WorkflowControlEndpoint, 479–483
tasks for implementing, 480
testing and, 482
WorkflowControlEndpoint class, 57, 443
WorkflowDesigner class, 754, 860
WorkflowExtensions property, 362, 405
WorkflowIdleBehavior class, 413, 443, 454
WorkflowInstanceQuery, 576
WorkflowInstanceRecord, 569, 576, 592
WorkflowInstanceStates class, 570
WorkflowInvoker class, 13, 49, 111–126, 267, 860
BeginInvoke method for, 113, 122, 124
EndInvoke method for, 113, 122, 124
executing activities via, 121
host transactions and, 547
instance methods for, 112, 122–126
Invoke method for, 111
InvokeAsync method for, 113, 122
static methods for, 111, 114–122
unhandled exceptions and, 507
workflow extensions and, 277
workflow persistence and, 416
vs. WorkflowApplication class, 126, 273
WorkflowItemPresenter class, 632, 651, 655, 660
WorkflowItemsPresenter class, 632, 655
WorkflowQueue (WF 3.x), migration and, 805
workflows, 1–44, 859
    aborting, 849
    activity data flow and, 99
    activity definition vs. activity instance, 55
    asynchronous execution and, 126
    bookmarks and, 131
    building blocks and, 97
    cancelling, 564, 851
    controlling workflow instances and, 132
    declarative activity model and, 45
    declarative workflow model and, 2
    declaring, 8–12, 17, 236
    dynamic updates, WF 3.x and, 68
    examples of. See samples
    executing, 13, 15, 111–161
    flowchart modeling for, 229–263
    hosting. See hosting workflows
    invoking from ASP.NET web applications, 145–149
    long-running, 265, 327, 479–483
    mental checklist for, 4
    multiple instances of, managing, 150–161
    orphaned workflow instances and, 443
    parallel execution, multiple activities and, 725–734
    passing parameters to, 17–22
    persistence and. See persistence
    procedural modeling for, 163–194
    reasons for using, 1
    resumption of after becoming idle, 113
    activities and. See activities; custom activities
    stopping execution of, 132, 137
    synchronous execution and, 148
    timeout values for, 119
    tracking. See tracking workflows
    transactions and, 411
    understanding as synonymous with activities, 46, 55
    unit testing for, 39–43
    updating, 508–564
    workflow extensions and, 129, 409
    Xaml workflow services and, 326
    Xaml workflows and, 326
WorkflowServiceHost
    workflow persistence and, 443–467
    WorkflowControlEndpoint and, 480
WorkflowServiceHost class, 49, 361, 443, 860
    Delay activity and, 423
    ExternalDataExchangeService and, 823, 831
    multiple workflow instances and, 150
    workflow persistence and, 416, 422
WorkflowExtensions property, 362, 405
Xamlx workflow services and, 326
WorkflowUnhandledExceptionBehavior class, 57, 412
WorkflowViewElement class, 630
WPF (Windows Presentation Foundation) applications, 131, 160, 629
WriteLine activity, 10, 103, 108
HostingDemoWorkflow sample and, 114
inventory lookup sample and, 188

X
x:Class attribute, 630, 636
Xaml, 15–17, 46, 860
   ActivityXamlServices class and, 142–144
   authoring activities and, 72
   breakpoints and, 38
Xaml workflows, 326
Xamlx workflow services, 326, 860
XML, serializing tracking records to, 605–608
xmlns:s attribute, 637
xmlns:sapc attribute, 637