**Windows Workflow Foundation Overview**

Windows Workflow Foundation is a framework that enables users to create system or human workflows in their applications written for Windows Vista, Windows XP, and the Windows Server 2003 operating systems. It consists of a namespace, an in-process workflow engine, and designers for Visual Studio 2005. Windows Workflow Foundation can be used to solve simple scenarios, such as showing UI controls based on user input, or complex scenarios encountered by large enterprises, such as order processing and inventory control. Windows Workflow Foundation comes with a programming model, a rehostable and customizable workflow engine, and tools for quickly building workflow-enabled applications on Windows.

Scenarios that Windows Workflow Foundation addresses include:

* Enabling workflow within line-of-business applications.
* User-interface page flows.
* Document-centric workflows.
* Human workflows.
* Composite workflows for service-oriented applications.
* Business rule-driven workflows.
* Workflows for systems management.

Windows Workflow Foundation provides a consistent and familiar development experience with other .NET Framework 3.0 technologies, such as Windows Communication Foundation and Windows Presentation Foundation. The Windows Workflow Foundation API provides full support for Visual Basic .NET and C#, a specialized workflow compiler, debugging within a workflow, a graphical workflow designer, and developing your workflow completely in code or in markup. Windows Workflow Foundation also provides an extensible model and designer to build custom activities that encapsulate workflow functionality for end users or for reuse across multiple projects.

The concepts in this section are inherent to Windows Workflow Foundation. They are briefly described here to provide a basic understanding. The Programming Guide section of this SDK provides implementation details and a deeper level of knowledge.

## In This Section

[Workflows Overview](http://msdn.microsoft.com/en-us/library/aa349006.aspx)

[Activities Overview](http://msdn.microsoft.com/en-us/library/aa349008.aspx)

[Services Overview](http://msdn.microsoft.com/en-us/library/aa348814.aspx)

[Compensation Overview](http://msdn.microsoft.com/en-us/library/aa349368.aspx)

[Local Communication and Correlation Overview](http://msdn.microsoft.com/en-us/library/aa348826.aspx)

[Persistence Overview](http://msdn.microsoft.com/en-us/library/aa349367.aspx)

[Tracking Overview](http://msdn.microsoft.com/en-us/library/aa349358.aspx)

[Serialization Overview](http://msdn.microsoft.com/en-us/library/aa349369.aspx)

[Workflow Changes Overview](http://msdn.microsoft.com/en-us/library/aa349354.aspx)

[Rules and Conditions Overview](http://msdn.microsoft.com/en-us/library/aa348711.aspx)

[Fault Handling Overview](http://msdn.microsoft.com/en-us/library/aa349370.aspx)

[Workflow Markup Overview](http://msdn.microsoft.com/en-us/library/aa349003.aspx)

**Workflows Overview**

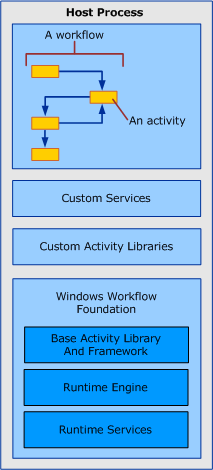
A workflow is a set of elemental units called activities that are stored as a model that describes a real-world process. Workflows provide a way of describing the order of execution and dependent relationships between pieces of short- or long-running work. This work passes through the model from start to finish, and activities might be executed by people or by system functions.

## Workflow Runtime Engine

Every running workflow instance is created and maintained by an in-process runtime engine that is commonly referred to as the workflow runtime engine. There can be several workflow runtime engines within an application domain, and each instance of the runtime engine can support multiple workflow instances running concurrently.

When a workflow model is compiled, it can be executed inside any Windows process including console applications, forms-based applications, Windows Services, ASP.NET Web sites, and Web services. Because a workflow is hosted in process, a workflow can easily communicate with its host application.

The following illustration shows how workflows, activities, and the workflow runtime engine are all hosted in process with a host application.



For more information about workflows, see [Developing Workflows](http://msdn.microsoft.com/en-us/library/ms734628.aspx) and [Developing Workflow-Enabled Applications](http://msdn.microsoft.com/en-us/library/ms735868.aspx).

**Activities Overview**

Activities are the elemental unit of a workflow. They are added to a workflow programmatically in a manner similar to adding XML DOM child nodes to a root node. When all the activities in a given flow path are finished running, the workflow instance is completed.

An activity can perform a single action, such as writing a value to a database, or it can be a composite activity and consist of a set of activities. Activities have two types of behavior: runtime and design time. The runtime behavior specifies the actions upon execution. The design-time behavior controls the appearance of the activity and its interaction while being displayed within the designer.

Windows Workflow Foundation contains a library of standard activities and provides the mechanisms for you to create your own. This enables extensibility and reusability between workflows.

For more information about standard activities, see [Windows Workflow Foundation Activities](http://msdn.microsoft.com/en-us/library/ms733615.aspx). For information about creating new activities, see [Developing Workflow Activities](http://msdn.microsoft.com/en-us/library/ms734413.aspx).

**Services Overview**

The workflow runtime engine uses many services when a workflow instance runs. Windows Workflow Foundation provides default implementations of the runtime services that meet the needs of many types of applications, such as a persistence service, which stores the execution details of a workflow instance in a SQL database. These service components are pluggable, which allows applications to provide these services in ways that are unique to their execution environment. Other types of services used by the runtime engine include scheduling services, transaction services, and tracking services.

Custom services can be created to extend the Windows Workflow Foundation platform by deriving from the base service classes. An example of this would be a persistence service that uses an XML file instead of a database for storage.

For more information about services, see [Windows Workflow Foundation Services](http://msdn.microsoft.com/en-us/library/ms734738.aspx) and [Developing Windows Workflow Foundation Services](http://msdn.microsoft.com/en-us/library/ms734705.aspx).

**Compensation Overview**

Compensation is the act of undoing any actions that were performed by a successfully completed compensatable activity because of an exception that occurred elsewhere in a workflow.

For more information about compensation, see [Using Compensation in Workflows](http://msdn.microsoft.com/en-us/library/ms735874.aspx) and [Using the CompensateActivity Activity](http://msdn.microsoft.com/en-us/library/ms735955.aspx).

**Local Communication and Correlation Overview**

Host processes can communicate with workflows by exchanging data through custom local communication services. These local communication services implement user-defined interfaces that define methods and events that will be passed between the workflow and the host process.

Host processes can also interact with a specific activity in a specific workflow instance by using a unique ID that is passed between the host process and the workflow as an event argument. This is known as correlation.

For more information about local communication and correlation, see [Workflow and Application Communication](http://msdn.microsoft.com/en-us/library/ms735924.aspx).

**Persistence Overview**

Windows Workflow Foundation simplifies the process of creating stateful, long-running, persistent workflow applications. The workflow runtime engine manages workflow execution and enables workflows to remain active for long periods of time and survive application restarts. This durability is a key tenet of Windows Workflow Foundation. It means that workflows can be unloaded from memory while awaiting input and serialized into a persistent store, such as a SQL database or XML file. Whenever the input is received, the workflow runtime engine loads the workflow state information back into memory and continues execution of the workflow.

Windows Workflow Foundation provides the [SqlWorkflowPersistenceService](http://msdn.microsoft.com/en-us/library/system.workflow.runtime.hosting.sqlworkflowpersistenceservice.aspx) that integrates well with Microsoft SQL Server 2005 Express, SQL Server 2000 or later, or SQL Server 2000 Desktop Engine (MSDE) to persist workflow information easily and efficiently. You can also create your own persistence service to store workflow state information any way you want by deriving from the [WorkflowPersistenceService](http://msdn.microsoft.com/en-us/library/system.workflow.runtime.hosting.workflowpersistenceservice.aspx) base class.

For more information about persistence, see [Windows Workflow Persistence Services](http://msdn.microsoft.com/en-us/library/ms734764.aspx) and [Creating Custom Persistence Services](http://msdn.microsoft.com/en-us/library/ms734700.aspx).

**Tracking Overview**

Tracking is the ability to specify and capture information about workflow instances and store that information as the instances execute. Windows Workflow Foundation provides the [SqlTrackingService](http://msdn.microsoft.com/en-us/library/system.workflow.runtime.tracking.sqltrackingservice.aspx), which is a tracking service that uses a SQL database to store the collected tracking information. You can also write your own tracking service to collect and store this information in any format that your application requires.

When a new workflow is created, the tracking service requests a tracking channel to be associated with that workflow. All of the tracking information from the workflow is then sent to this tracking channel.

The tracking service can track three types of events: Workflow instance events, Activity events, and User events. You can configure the type and amount of information that your service wants to receive for a particular workflow instance or type of workflow by providing a tracking profile.

The tracking framework also provides the ability to extract information about activities or the workflow during an event. If a specific property or field in your activity or workflow needs to be tracked, you can provide this information in the extracts section of the tracking profile, and that information will be extracted during the specified event.

For more information about tracking, see [Windows Workflow Tracking Services](http://msdn.microsoft.com/en-us/library/ms735887.aspx), **SqlTrackingService**, and [Creating Custom Tracking Services](http://msdn.microsoft.com/en-us/library/ms735912.aspx).

**Serialization Overview**

Workflows, activities, and rules can be serialized and deserialized. This enables you to persist them, use them in workflow markup files, and view their properties, fields, and events in a workflow designer.

Windows Workflow Foundation provides default serialization capabilities for standard activities, or you can create your own for custom activities. For example, with a custom activity serializer, you can decide which members are serialized and how they are serialized. This determines if those members are visible or hidden in a workflow designer.

For more information about serialization, see [How to: Serialize Workflows](http://msdn.microsoft.com/en-us/library/ms735702.aspx) and [Serializing Custom Activities](http://msdn.microsoft.com/en-us/library/ms735923.aspx).

**Workflow Changes Overview**

Windows Workflow Foundation enables you to dynamically update your workflow instance and declarative rules during run time. Before activities are scheduled for execution, you can change expected behaviors, flow control, and so on. This ability enables you to modify business processing logic without having to recompile and restart your workflow.

For more information about workflow changes, see [Using Workflow Changes in Workflows](http://msdn.microsoft.com/en-us/library/ms734775.aspx) and [How to: Apply Workflow Changes to Workflows](http://msdn.microsoft.com/en-us/library/ms734569.aspx).

**Rules and Conditions Overview**

Windows Workflow Foundation can implement business logic as either rules or conditions. Conditions are used by [IfElseBranchActivity](http://msdn.microsoft.com/en-us/library/system.workflow.activities.ifelsebranchactivity.aspx), [ConditionedActivityGroup](http://msdn.microsoft.com/en-us/library/system.workflow.activities.conditionedactivitygroup.aspx), [WhileActivity](http://msdn.microsoft.com/en-us/library/system.workflow.activities.whileactivity.aspx), and [ReplicatorActivity](http://msdn.microsoft.com/en-us/library/system.workflow.activities.replicatoractivity.aspx) activities to control activity execution. Conditions can be expressed as declarative, or defined in code. Declarative conditions are created as code DOM statements in the rules' XML file. Code-based conditions reference a method in the workflow's code file that returns its result through the [Result](http://msdn.microsoft.com/en-us/library/system.workflow.activities.conditionaleventargs.result.aspx) property.

Rules, like conditions, are expressed as code DOM statements, and are collected in the rules XML file. Rules include a condition statement and collections of actions that are performed based on the result of the condition. Rules are collected into rule sets, which support both simple sequential execution of rules, and sophisticated forward-chaining of rules. Rule sets are executed by the [PolicyActivity](http://msdn.microsoft.com/en-us/library/system.workflow.activities.policyactivity.aspx) activity.

A key advantage of defining your logic with rules and declarative conditions is that they can be modified at run time by doing dynamic updates using workflow changes. In addition, rules let you separate your business logic from a workflow in order to share those rules with other workflows. Finally, defining business logic in rules allows for advanced tools, such as dependency visualization and impact analysis tools, to be built on top of the object model.

For more information about conditions, see [Using Conditions in Workflows](http://msdn.microsoft.com/en-us/library/ms734614.aspx); for information about rules, see [Using RuleSets in Workflows](http://msdn.microsoft.com/en-us/library/aa348818.aspx).

**Fault Handling Overview**

Exceptions that occur in activities are handled asynchronously by the workflow runtime engine in a process called fault handling. Exceptions are scheduled in a queue to be handled later. If the exception type matches the one that is handled by a particular [FaultHandlerActivity](http://msdn.microsoft.com/en-us/library/system.workflow.componentmodel.faulthandleractivity.aspx) activity, that activity will handle the exception. If the exception cannot be handled, it is bubbled up through parent activities until it ultimately causes the termination of the workflow instance.

For more information about fault handling, see [Fault Handling in Workflows](http://msdn.microsoft.com/en-us/library/ms734771.aspx).

**Workflow Markup Overview**

Workflow markup, which is based on extensible Application Markup Language (XAML), enables developers and designers to model business logic declaratively and separate it from lower-level implementation details that are modeled by code-beside files. Because workflows can be modeled declaratively, it is possible to activate a workflow by directly loading a workflow markup file into the workflow runtime engine at run time.

For more information about workflow markup, see [Using Workflow Markup](http://msdn.microsoft.com/en-us/library/ms735921.aspx).