**Hosting Services**

To become active, a service must be hosted within a run-time environment that creates it and controls its context and lifetime. Windows Communication Foundation (WCF) services are designed to run in any Windows process that supports managed code.

WCF provides a unified programming model for building service-oriented applications. This programming model remains consistent and is independent of the run-time environment in which the service is deployed. In practice, this means that the code for your services looks much the same whatever the hosting option.

These hosting options range from running inside a console application to server environments such as a Windows service running within a worker process managed by Internet Information Services (IIS) or by Windows Process Activation Service (WAS). Developers choose the hosting environment that satisfies the service's deployment requirements. These requirements might derive from the platform on which the application is deployed, the transport on which it must send and receive messages, or on the type of process recycling and other process management required to ensure adequate availability, or on some other management or reliability requirements. The next section provides information and guidance on hosting options.

**Hosting Options**

**Self-Hosting in a Managed Application**

WCF services can be hosted in any managed application. This is the most flexible option because it requires the least infrastructure to deploy. You embed the code for the service inside the managed application code and then create and open an instance of the [ServiceHost](http://msdn.microsoft.com/en-us/library/system.servicemodel.servicehost.aspx) to make the service available. For more information, see [How to: Host a WCF Service in a Managed Application](http://msdn.microsoft.com/en-us/library/ms731758.aspx).

This option enables two common scenarios: WCF services running inside console applications and rich client applications such as those based on Windows Presentation Foundation (WPF) or Windows Forms (WinForms). Hosting a WCF service inside a console application is typically useful during the application's development phase. This makes them easy to debug, easy to get trace information from to find out what is happening inside of the application, and easy to move around by copying them to new locations. This hosting option also makes it easy for rich client applications, such as WPF and WinForms applications, to communicate with the outside world. For example, a peer-to-peer collaboration client that uses WPF for its user interface and also hosts a WCF service that allows other clients to connect to it and share information.

**Managed Windows Services**

This hosting option consists of registering the application domain (AppDomain) that hosts an WCF service as a managed Windows Service (formerly known as NT service) so that the process lifetime of the service is controlled by the service control manager (SCM) for Windows services. Like the self-hosting option, this type of hosting environment requires that some hosting code is written as part of the application. The service is implemented as both a Windows Service and as an WCF service by causing it to inherit from the [ServiceBase](http://msdn.microsoft.com/en-us/library/system.serviceprocess.servicebase.aspx) class as well as from an WCF service contract interface. The **ServiceHost** is then created and opened within an overridden [OnStart](http://msdn.microsoft.com/en-us/library/system.serviceprocess.servicebase.onstart.aspx) method and closed within an overridden [OnStop](http://msdn.microsoft.com/en-us/library/system.serviceprocess.servicebase.onstop.aspx) method. An installer class that inherits from [Installer](http://msdn.microsoft.com/en-us/library/system.configuration.install.installer.aspx) must also be implemented to allow the program to be installed as a Windows Service by the Installutil.exe tool. For more information, see [How to: Host a WCF Service in a Managed Windows Service](http://msdn.microsoft.com/en-us/library/ms733069.aspx). The scenario enabled by the managed Windows Service hosting option is that of a long-running WCF service hosted outside of IIS in a secure environment that is not message-activated. The lifetime of the service is controlled instead by the operating system. This hosting option is available in all versions of Windows.

**Internet Information Services (IIS)**

The IIS hosting option is integrated with ASP.NET and uses the features these technologies offer, such as process recycling, idle shutdown, process health monitoring, and message-based activation. On the Windows XP and Windows Server 2003 operating systems, this is the preferred solution for hosting Web service applications that must be highly available and highly scalable. IIS also offers the integrated manageability that customers expect from an enterprise-class server product. This hosting option requires that IIS be properly configured, but it does not require that any hosting code be written as part of the application. For more information about how to configure IIS hosting for a WCF service, see [How to: Host a WCF Service in IIS](http://msdn.microsoft.com/en-us/library/ms733766.aspx).

Note that IIS-hosted services can only use the HTTP transport. Its implementation in IIS 5.1 has introduced some limitations in Windows XP. The message-based activation provided for an WCF service by IIS 5.1 on Windows XP blocks any other self-hosted WCF service on the same computer from using port 80 to communicate. WCF services can run in the same AppDomain/Application Pool/Worker Process as other applications when hosted by IIS 6.0 on Windows Server 2003. But because WCF and IIS 6.0 both use the kernel-mode HTTP stack (HTTP.sys), IIS 6.0 can share port 80 with other self-hosted WCF services running on the same machine, unlike IIS 5.1.

**Windows Process Activation Service (WAS)**

Windows Process Activation Service (WAS) is the new process activation mechanism for the Windows Server 2008 that is also available on Windows Vista. It retains the familiar IIS 6.0 process model (application pools and message-based process activation) and hosting features (such as rapid failure protection, health monitoring, and recycling), but it removes the dependency on HTTP from the activation architecture. IIS 7.0 uses WAS to accomplish message-based activation over HTTP. Additional WCF components also plug into WAS to provide message-based activation over the other protocols that WCF supports, such as TCP, MSMQ, and named pipes. This allows applications that use communication protocols to use the IIS features such as process recycling, rapid fail protection, and the common configuration system that were only available to HTTP-based applications.

This hosting option requires that WAS be properly configured, but it does not require you to write any hosting code as part of the application. For more information about how to configure WAS hosting, see [How to: Host a WCF Service in WAS](http://msdn.microsoft.com/en-us/library/ms733109.aspx).

**Choosing a Hosting Environment**

The following table summarizes some of the key benefits and scenarios associated with each of the hosting options.

|  |  |  |
| --- | --- | --- |
| **Hosting Environment** | **Common Scenarios** | **Key Benefits and Limitations** |
| Managed Application ("Self-Hosted") | * Console applications used during development. * Rich WinForm and WPF client applications accessing services. | * Flexible. * Easy to deploy. * Not an enterprise solution for services. |
| Windows Services (formerly known as NT services) | * A long-running WCF service hosted outside of IIS. | * Service process lifetime controlled by the operating system, not message-activated. * Supported by all versions of Windows. * Secure environment. |
| IIS 5.1, IIS 6.0 | * Running a WCF service side-by-side with ASP.NET content on the Internet using the HTTP protocol. | * Process recycling. * Idle shutdown. * Process health monitoring. * Message-based activation. * HTTP only. |
| Windows Process Activation Service (WAS) | * Running a WCF service without installing IIS on the Internet using various transport protocols. | * IIS is not required. * Process recycling. * Idle shutdown. * Process health monitoring. * Message-based activation. * Works with HTTP, TCP, named pipes, and MSMQ. |
| IIS 7.0 | * Running a WCF service with ASP.NET content. * Running a WCF service on the Internet using various transport protocols. | * WAS benefits. * Integrated with ASP.NET and IIS content. |

The choice of a hosting environment depends on the version of Windows on which it is deployed, the transports it requires to send messages and the type of process and application domain recycling it requires. The following table summarizes the data related to these requirements.

|  |  |  |  |
| --- | --- | --- | --- |
| **Hosting Environment** | **Platform Availability** | **Transports Supported** | **Process and AppDomain Recycling** |
| Managed Applications ("Self-Hosted") | Windows XP, Windows Server 2003, Windows Vista,  Windows Server 2008 | HTTP,  net.tcp,  net.pipe,  net.msmq | No |
| Windows Services (formerly known as NT services) | Windows XP, Windows Server 2003, Windows Vista,  Windows Server 2008 | HTTP,  net.tcp,  net.pipe,  net.msmq | No |
| IIS 5.1 | Windows XP | HTTP | Yes |
| IIS 6.0 | Windows Server 2003 | HTTP | Yes |
| Windows Process Activation Service (WAS) | Windows Vista, Windows Server 2008 | HTTP,  net.tcp,  net.pipe,  net.msmq | Yes |

It is important to note that running a service or any extension from an untrusted host compromises security. Also, note that when opening a **ServiceHost** under impersonation, an application must ensure that the user is not logged off, for example by caching the [WindowsIdentity](http://msdn.microsoft.com/en-us/library/system.security.principal.windowsidentity.aspx) of the user.