**.NET Framework Remoting Overview**

This page is specific to

**.NET Framework 3.0**

Other versions are also available for the following:

[Microsoft Visual Studio 2003/.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/kwdt6w2k(VS.71).aspx)

[Microsoft Visual Studio 2005/.NET Framework 2.0](http://msdn.microsoft.com/en-us/library/kwdt6w2k(VS.80).aspx)

**This topic is specific to a legacy technology that is retained for backward compatibility with existing applications and is not recommended for new development. Distributed applications should now be developed using the** [Windows Communication Foundation (WCF)](http://go.microsoft.com/fwlink/?LinkID=127777).

.NET remoting enables you to build widely distributed applications easily, whether the application components are all on one computer or spread out across the entire world. You can build client applications that use objects in other processes on the same computer or on any other computer that is reachable over its network. You can also use .NET remoting to communicate with other application domains in the same process. (For details about programming application domains, see [Programming with Application Domains](http://msdn.microsoft.com/en-us/library/yk22e11a(VS.85).aspx).)

.NET remoting provides an abstract approach to interprocess communication that separates the remotable object from a specific client or server application domain and from a specific mechanism of communication. As a result, it is flexible and easily customizable. You can replace one communication protocol with another, or one serialization format with another without recompiling the client or the server. In addition, the remoting system assumes no particular application model. You can communicate from a Web application, a console application, a Windows Service – from almost anything you want to use. Remoting servers can also be any type of application domain. Any application can host remoting objects and provide its services to any client on its computer or network.

|  |
| --- |
| **NoteNote** |
| For security reasons, it is strongly recommended to expose Remoting endpoints through secure channels. Never expose insecure Remoting endpoints to the Internet. |

To use .NET remoting to build an application in which two components communicate directly across an application domain boundary, you need to build only the following:

* A remotable object.
* A host application domain to listen for requests for that object.
* A client application domain that makes requests for that object.

Even in a complex, multiclient or multiserver application, .NET remoting can be thought of in this way. The host and the client application must also be configured with the remoting infrastructure and you must understand the lifetime and activation issues that the remoting infrastructure introduces.

**// remoting / RemotableType.cs**

using System;

public class RemotableType : MarshalByRefObject

{

public string SayHello()

{

Console.WriteLine("RemotableType.SayHello() was called!");

return "Hello, world";

}

}

**// remoting / listener/ Listener.cs**

using System;

using System.Runtime.Remoting;

public class Listener

{

public static void Main(string[] args)

{

RemotingConfiguration.Configure("Listener.exe.config", false);

Console.WriteLine("Listening for requests. Press enter to exit...");

Console.ReadLine();

}

}

**// remoting / listener/ listener.exe.config**

<configuration>

<system.runtime.remoting>

<application>

<service>

<wellknown

mode="Singleton"

type="RemotableType, RemotableType"

objectUri="RemotableType.rem"

/>

</service>

<channels>

<channel ref="http" port="8989"/>

</channels>

</application>

</system.runtime.remoting>

</configuration>

**// remoting / client/ Client.cs**

using System;

using System.Runtime.Remoting;

public class Client{

public static void Main(){

RemotingConfiguration.Configure("Client.exe.config");

RemotableType remoteObject = new RemotableType();

Console.WriteLine(remoteObject.SayHello());

}

}

**// remoting / client/ client.exe.config**

<configuration>

<system.runtime.remoting>

<application>

<client>

<wellknown

type="RemotableType, RemotableType"

url="http://localhost:8989/RemotableType.rem"

/>

</client>

</application>

</system.runtime.remoting>

</configuration>

**To compile and run a basic remoting application**

1. At the command prompt in the remoting\type directory, type the following command: **csc /noconfig /t:library RemotableType.cs**
2. Copy RemotableType.dll into the remoting\client and remoting\listener directories.
3. Copy listener.exe.config into the remoting\listener directory.
4. Copy client.exe.config into the remoting\client directory.
5. At the command-prompt in the remoting\listener directory, type the following command: **csc /noconfig /r:RemotableType.dll Listener.cs**
6. At the command-prompt in the remoting\client directory, type the following command: **csc /noconfig /r:RemotableType.dll Client.cs**
7. At the command prompt in the remoting\listener directory, type Listener.
8. When the Listener application is running, open a new command prompt in the remoting\client directory and type Client.