**Remoting, Web Services**

#### Remoting

#### How do in-process and cross-process communication work in the Common Language Runtime?

There are two aspects to in-process communication: between contexts within a single application domain, or across application domains. Between contexts in the same application domain, proxies are used as an interception mechanism. No marshaling/serialization is involved. When crossing application domains, we do marshaling/serialization using the runtime binary protocol.

Cross-process communication uses a pluggable channel and formatter protocol, each suited to a specific purpose.

* If the developer specifies an endpoint using the tool soapsuds.exe to generate a metadata proxy, HTTP channel with SOAP formatter is the default.
* If a developer is doing explicit remoting in the managed world, it is necessary to be explicit about what channel and formatter to use. This may be expressed administratively, through configuration files, or with API calls to load specific channels. Options are:

HTTP channel w/ SOAP formatter (HTTP works well on the Internet, or anytime traffic must travel through firewalls)

TCP channel w/ binary formatter (TCP is a higher performance option for local-area networks (LANs))

When making transitions between managed and unmanaged code, the COM infrastructure (specifically, DCOM) is used for remoting. In interim releases of the CLR, this applies also to serviced components (components that use COM+ services). Upon final release, it should be possible to configure any remotable component.

Distributed garbage collection of objects is managed by a system called "leased based lifetime." Each object has a lease time, and when that time expires, the object is disconnected from the remoting infrastructure of the CLR. Objects have a default renew time-the lease is renewed when a successful call is made from the client to the object. The client can also explicitly renew the lease.

**What’s a Windows process?**

It’s an application that’s running and had been allocated memory.

**What’s typical about a Windows process in regards to memory allocation?**

Each process is allocated its own block of available RAM space; no process can access another process’ code or data. If the process crashes, it dies alone without taking the entire OS or a bunch of other applications down.

**Why do you call it a process? What’s different between process and application in .NET, not common computer usage, terminology?**

A process is an instance of a running application. An application is an executable on the hard drive or network. There can be numerous processes launched of the same application (5 copies of Word running), but 1 process can run just 1 application.

**What distributed process frameworks outside .NET do you know?**

Distributed Computing Environment/Remote Procedure Calls (DEC/RPC), Microsoft Distributed Component Object Model (DCOM), Common Object Request Broker Architecture (CORBA), and Java Remote Method Invocation (RMI).

**What are possible implementations of distributed applications in .NET?**

.NET Remoting and ASP.NET Web Services. If we talk about the Framework Class Library, noteworthy classes are in System.Runtime.Remoting and System.Web.Services.

**When would you use .NET Remoting and when Web services?**

Use remoting for more efficient exchange of information when you control both ends of the application. Use Web services for open-protocol-based information exchange when you are just a client or a server with the other end belonging to someone else.

**What’s a proxy of the server object in .NET Remoting?**

It’s a fake copy of the server object that resides on the client side and behaves as if it was the server. It handles the communication between real server object and the client object. This process is also known as **marshaling**.

**What are remotable objects in .NET Remoting?**

Remotable objects are the objects that can be marshaled across the application domains. You can marshal by value, where a deep copy of the object is created and then passed to the receiver. You can also marshal by reference, where just a reference to an existing object is passed.

**What are channels in .NET Remoting?**

Channels represent the objects that transfer the other serialized objects from one application domain to another and from one computer to another, as well as one process to another on the same box. A channel must exist before an object can be transferred.

**What security measures exist for .NET Remoting in System.Runtime.Remoting?**

None. Security should be taken care of at the application level. Cryptography and other security techniques can be applied at application or server level.

**What is a formatter?**

A formatter is an object that is responsible for encoding and serializing data into messages on one end, and deserializing and decoding messages into data on the other end.

**Choosing between HTTP and TCP for protocols and Binary and SOAP for formatters, what are the trade-offs?**

Binary over TCP is the most efficient, SOAP over HTTP is the most interoperable.

**What’s SingleCall activation mode used for?**

If the server object is instantiated for responding to just one single request, the request should be made in SingleCall mode.

**What’s Singleton activation mode?**

A single object is instantiated regardless of the number of clients accessing it. Lifetime of this object is determined by lifetime lease.

**How do you define the lease of the object?**

By implementing ILease interface when writing the class code.

**Can you configure a .NET Remoting object via XML file?**

Yes, via machine.config and application level .config file (or web.config in ASP.NET). Application-level XML settings take precedence over machine.config.

**How can you automatically generate interface for the remotable object in .NET with Microsoft tools?**

Use the Soapsuds tool.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**What’s a Windows process?** It’s an application that’s running and had been allocated memory.

**What’s typical about a Windows process in regards to memory allocation?** Each process is allocated its own block of available RAM space, no process can access another process’ code or data. If the process crashes, it dies alone without taking the entire OS or a bunch of other applications down.

**Why do you call it a process? What’s different between process and application in .NET, not common computer usage, terminology?** A process is an instance of a running application. An application is an executable on the hard drive or network. There can be numerous processes launched of the same application (5 copies of Word running), but 1 process can run just 1 application.

**What distributed process frameworks outside .NET do you know?** Distributed Computing Environment/Remote Procedure Calls (DEC/RPC), Microsoft Distributed Component Object Model (DCOM), Common Object Request Broker Architecture (CORBA), and Java Remote Method Invocation (RMI).

**What are possible implementations of distributed applications in .NET?** .NET Remoting and ASP.NET Web Services. If we talk about the Framework Class Library, noteworthy classes are in System.Runtime.Remoting and System.Web.Services.

**When would you use .NET Remoting and when Web services?** Use remoting for more efficient exchange of information when you control both ends of the application. Use Web services for open-protocol-based information exchange when you are just a client or a server with the other end belonging to someone else.

**What’s a proxy of the server object in .NET Remoting?** It’s a fake copy of the server object that resides on the client side and behaves as if it was the server. It handles the communication between real server object and the client object. This process is also known as **marshaling**.

**What are remotable objects in .NET Remoting?** Remotable objects are the objects that can be marshaled across the application domains. You can marshal by value, where a deep copy of the object is created and then passed to the receiver. You can also marshal by reference, where just a reference to an existing object is passed.

**What are channels in .NET Remoting?** Channels represent the objects that transfer the other serialized objects from one application domain to another and from one computer to another, as well as one process to another on the same box. A channel must exist before an object can be transferred.

**What security measures exist for .NET Remoting in System.Runtime.Remoting?** None. Security should be taken care of at the application level. Cryptography and other security techniques can be applied at application or server level.

**What is a formatter?** A formatter is an object that is responsible for encoding and serializing data into messages on one end, and deserializing and decoding messages into data on the other end.

**Choosing between HTTP and TCP for protocols and Binary and SOAP for formatters, what are the trade-offs?** Binary over TCP is the most effiecient, SOAP over HTTP is the most interoperable.

**What’s SingleCall activation mode used for?** If the server object is instantiated for responding to just one single request, the request should be made in SingleCall mode.

**What’s Singleton activation mode?** A single object is instantiated regardless of the number of clients accessing it. Lifetime of this object is determined by lifetime lease.

**How do you define the lease of the object?** By implementing ILease interface when writing the class code.

**Can you configure a .NET Remoting object via XML file?** Yes, via machine.config and application level .config file (or web.config in ASP.NET). Application-level XML settings take precedence over machine.config.

**How can you automatically generate interface for the remotable object in .NET with Microsoft tools?** Use the Soapsuds tool.

**What is the transport protocol you use to call a Web service?**SOAP (Simple Object Access Protocol) is the preferred protocol. 

**True or False: A Web service can only be written in .NET?**False

**What does WSDL stand for?**Web Services Description Language. 

**Where on the Internet would you look for Web services?**[http://www.uddi.org](http://www.uddi.org/) 

**True or False: To test a Web service you must create a Windows application or Web application to consume this service?**False, the web service comes with a test page and it provides HTTP-GET method to test.

Webservices are created with the following two files

1. .asmx
2. .cs
3. html
4. **1 and 2**

Is overloading possible in web services?

Yes, We can overload webmethods in a webservice.   
  
There is the MessageName property of a WebMethod attribute! The MessageName property enables the XML Web Service to uniquely identify overloaded methods using an alias. The default value of this property is the method name, but you can change it to distinguish this overloaded method implementation from others. When specifying the method name, the resulting SOAP messages will reflect this name instead of an actual method name.

Can two different programming languages be mixed in a single ASMX file?

NO.

What is singlecall?

Single Call types always have one instance per client request. The next method invocation will be serviced by a different server instance, even if the previous instance has not yet been recycled by the system.

What is singleton?

Singleton types never have more than one instance at any one time. If an instance exists, all client requests are serviced by that instance.

**What is Remoting?**

The process of communication between different operating system processes, regardless of whether they are on the same computer. The .NET remoting system is an architecture designed to simplify communication between objects living in different application domains, whether on the same computer or not, and between different contexts, whether in the same application domain or not.

How to Create And Delploy A WebService Using C#

First, let's start off by creating a very simple webservice.   
Creating A Webservice   
1.Create a folder named Webservice under wwwroot   
2. Create a File   
<%@ WebService Language="c#" Class="AddNumbers"%>   
  
using System;   
using System.Web.Services;   
public class AddNumbers : WebService   
{   
[WebMethod]   
public int Add(int a, int b){   
int sum;   
sum = a + b;   
return sum;   
}   
}   
3.Save this file as AddService.asmx [asmx-> file extension]   
4.Now the webservice is created and ready for the clients to use it.   
5. Now we can call this webservice using   
http://ip address/Webservice/Addservice.asmx/Add?a=10&b=5   
This will return the result in XML format   
Deploying the Webservice in the Client Machine   
1.At the command prompt:   
WSDL http://ip address ofthe site/WebService/MathService.asmx /n:NameSp /out:FileName.cs]   
-This will create a file called FileNmame.cs .   
WSDL -> WebServices Description Language (This is an application available at C:\Program   
Files\Microsoft.NET\FrameworkSDK\Bin)   
NameSp -> Name of the NameSpace which will be used in client code for deploying the webservice.   
2.Compilation   
CSC /t:library /r:system.web.dll /r:system.xml.dll CreatedFile.cs   
This will create a dll with the name of the public class of the asmx file.( In our case, it is "AddNumbers.dll" )   
CSC is an application available at C:\WINNT\Microsoft.NET\Framework\v1.0.2914   
3.Put the dll file inside WWWRooT\BIN [Create a BIN Folder in WWWRoot]   
Making use of WebService in client asp/aspx page   
<%@ import Namespace = "NameSp" %>   
<script language = "c#" runat = "server">   
public void Page\_Load(object o, EventArgs e){   
int x = 10;   
int y = 5;   
int sum;   
//Instantiating the public class of the webservice   
AddNumbers AN = new AddNumbers();   
sum = AN.Add(x,y);   
string str = sum.ToString();   
response.writeline(str);   
}   
</script>   
Note   
It is advisable to   
1. Copy the .asmx file to the folder containing WSDL aplication (C:\Program Files\Microsoft.NET\FrameworkSDK\Bin) before creating cs file.   
2. Copy the created .cs file to the folder containing CSC application

When would you use .NET Remoting and when Web services?

Use remoting for more efficient exchange of information when you control both ends of the application. Use Web services for open-protocol-based information exchange when you are just a client or a server with the other end belonging to someone else.

Describe the term Channel in .Net Remoting

In .Net Remoting, an application use Channel to send message to another application which is running in different domain or process. Before sending message, Channel converts message into appropriate format like XML or binary format. The channel that carries message(Mashalled parameter) can use protocal like TCP and HTTP. Channel can be HTTPChannel and TCPChannel. The HTTPChannel use soapFormatter to serialize messages into the XML format using SOAP protocol. Using SOAP method allows the client to call method on the remote object that might not be using .Net framework. The TCPChannel use binaryFormatter to serialize message into binary stream.

What is UDDI?

UDDI is a platform-independent framework for describing services, discovering businesses, and integrating business services by using the Internet.   
  
1. UDDI stands for Universal Description, Discovery and Integration   
2. UDDI is a directory for storing information about web services   
3. UDDI is a directory of web service interfaces described by WSDL   
4. UDDI communicates via SOAP   
5. UDDI is built into the Microsoft .NET platform

What are the considerations in deciding to use .NET Remoting or ASP.NET Web Services?

Remoting is a more efficient communication exchange when you can control both ends of the application involved in the communication process. Web Services provide an open-protocol-based exchange of informaion. Web Services are best when you need to communicate with an external organization or another (non-.NET) technology.

What are the remotable and non-remotable objects in .Net Remoting?

The remotable objects are the objects which can be distributed accross domains, can be used with domain. The non-remotable objects are the objects which can't be distributed accross domains. In distributed system, if an object is very big, we can make it non-remotable object.

Describe the type of remotable objects in .Net Remoting.

* Marshal-by-value-objects - When client calls a method on marshal-by-value-object, the remoting system creates a copy of this object and passes the copy to the client application domain. The copy hence received can handle any method call in client domain. Using Marshal-by-value-object reduces resource consuming trip across network.
* Marshal-by-reference-object - When client calls a method on Marshal by reference object, the remoting system create proxy object in the caller application that contains the reference of all method and properties of the object.

Describe the term Channel in .Net Remoting

In .Net Remoting, an application use Channel to send message to another application which is runing in different domain or process. Before sending message, Channel converts message into appropriate format like XML or binary format. The channel that carries message(Mashalled parameter) can use protocal like TCP and HTTP. Channel can be HTTPChannel and TCPChannel. The HTTPChannel use soapFormatter to serialize messages into the XML format using SOAP protocal. Using SOAP method allows the client to call method on the remote object that might not be using .Net framework. The TCPChannel use binaryFormatter to serialize message into binary stream.

**What is WSDL?**

WSDL (Web Services Description Language) is an XML-based language for describing Web services and how to access them.

**What security measures exist for .NET Remoting in System.Runtime.Remoting?**

None. Security should be taken care of at the application level. Cryptography and other security techniques can be applied at application or server level.

What’s Singleton activation mode?

A single object is instantiated regardless of the number of clients accessing it. Lifetime of this object is determined by lifetime lease.

What’s SingleCall activation mode used for?

If the server object is instantiated for responding to just one single request, the request should be made in SingleCall mode.

What is a formatter?

A formatter is an object that is responsible for encoding and serializing data into messages on one end, and deserializing and decoding messages into data on the other end.

What are remotable objects in .NET Remoting?

Remotable objects are the objects that can be marshaled across the application domains. You can marshal by value, where a deep copy of the object is created and then passed to the receiver. You can also marshal by reference, where just a reference to an existing object is passed.

**What is SOAP?**

SOAP (Simple Object Access Protocol) is the union of XML and HTTP. SOAP is W3C submitted note that uses XML and HTTP to encode & transmit application data.

What’s a proxy of the server object in .NET Remoting?

It’s a fake copy of the server object that resides on the client side and behaves as if it was the server. It handles the communication between real server object and the client object. This process is also known as marshaling.

What are the consideration in deciding to use .NET Remoting or ASP.NET Web Services?

Remoting is a more efficient communication exchange when you can control both ends of the application involved in the communication process. Web Services provide an open-protocol-based exchange of informaion. Web Services are best when you need to communicate with an external organization or another (non-.NET) technology.

What are channels in .NET Remoting?

Channels represent the objects that transfer the other serialized objects from one application domain to another and from one computer to another, as well as one process to another on the same box. A channel must exist before an object can be transferred.

What security measures exist for .NET Remoting in System.Runtime.Remoting?

None. Security should be taken care of at the application level. Cryptography and other security techniques can be applied at application or server level.

What is a formatter?

A formatter is an object that is responsible for encoding and serializing data into messages on one end, and deserializing and decoding messages into data on the other end.

Can you configure a .NET Remoting object via XML file?

Yes, via machine.config and application level .config file (or web.config in ASP.NET). Application-level XML settings take precedence over machine.config.

Choosing between HTTP and TCP for protocols and Binary and SOAP for formatters, what are the trade-offs?

Binary over TCP is the most effiecient, SOAP over HTTP is the most interoperable.

How do you define the lease of the object?

By implementing ILease interface when writing the class code.

What’s SingleCall activation mode used for?

If the server object is instantiated for responding to just one single request, the request should be made in SingleCall mode.

**what is the diff between remoting and webservice**

1. Web service is to achieve cross-platform interoperability where as .NET Remoting desgined expressly for .NET -to- .NET communications.

2. .NET Remoting would offer best performance for communicating .NET applications where as web services wouldn't in this case.

**what are the various type of channels used by .Net remoting and which is the best one?**

There are 2 types of channels .

channel means sending messages. 2 types are http and tcp

Tcp is used for lan , http is used for lan and wan

**In which conditions do you opt for Remoting services?**

Remoting is something like web services. You might decide to opt for

Remoting instead of web services in following cases...

1) Client and server platform is fixed

2) Protocol is NOT fixed. (Like web services strictly work

on HTTP protocol)

3) Where object serialization is to be done strictly

through CLR.

4) Where platform is fixed i.e. .Net framework.

when both the server and client r under our control then we

use .net remoting.

Remoting can be prefered

1. Both server and Client sit in different machine.

2. Both server and client are .NET applications

3. The state of the objects transfered accross the network

should be maintained.

4. The objects of the IDictionary interface should be

transfered accross the network.

**What are the differences between Marshal by value and Marshal by reference?**

Marshal-by-value objects are copied by the remoting system and passed in their entirety to the caller's application domain. Once copied to the caller's application domain (by the marshaling process), all method calls and property accesses are executed entirely within that domain. The entire object exists in the caller's domain, so there is no need to marshal accesses across domain boundaries. Using

marshal-by-value objects can increase performance and reduce network traffic when used for small objects or objects to which you will be making many accesses. However, because the object exists entirely in the caller's application domain, no state changes to the object are communicated to the originating application domain, or from the originator back to the caller. Marshal-by-value is not

a good choice for very large objects with many accesses. It makes little sense to marshal an entire large object across domain boundaries when all you need is access to a single field, method, or property.

Remoting makes an object in one process (the server) available to code in another process (the client). This is called marshalling, and there are two fundamentally different ways to marshal an object

-->Marshal by value: the server creates a copy of the

object passes the copy to the client.

-->Marshal by reference: the client creates a proxy for the

object and then uses the proxy to access the object.

**Choosing between HTTP and TCP for protocols and Binary and SOAP for formatters, what are the trade-offs? In what way it is best?**

Binary over TCP is the most effiecient, SOAP over HTTP is the most interoperable.

Soap formatters are preffered over Binary as they do not firewall issue.

**Can you configure a .NET Remoting object via XML file?**

Yes, via machine.config and application level .config file (or web.config in ASP.NET). Application-level XML settings take precedence over machine.config.

**What is Distributed computing?**

Distributed computing is an integral part of almost every software development. Before .Net Remoting, DCOM was the most used method of developing distributed application on Microsoft platform. Because of object oriented architecture, .NET Remoting replaces DCOM as .Net framework replaces COM.

**What are the benefits of Distributed Application Development?**

*Fault Tolerance:* Fault tolerance means that a system should be resilient when failures within the system occur.

*Scalability:* Scalability is the ability of a system to handle increased load with only an incremental change in performance.

*Administration:* Managing the system from one place.

In brief, .NET remoting is an architecture which enables communication between different application domains or processes using different transportation protocols, serialization formats, object lifetime schemes, and modes of object creation. Remote means any object which executes outside the application domain. The two processes can exist on the same computer or on two computers connected by a LAN or the Internet. This is called marshalling (This is the process of passing parameters from one context to another.), and there are two basic ways to marshal an object:

*Marshal by value:* the server creates a copy of the object and passes the copy to the client.

*Marshal by reference:* the client creates a proxy for the object and then uses the proxy to access the object.

**Web Services**

**What are Web Services?**

* Web services are application components
* Web services communicate using open protocols
* Web services are self-contained and self-describing
* Web services can be discovered using UDDI
* Web services can be used by other applications
* XML is the basis for Web services

**How does it Work?**

The basic Web services platform is XML + HTTP.

XML provides a language which can be used between different platforms and programming languages and still express complex messages and functions.

The HTTP protocol is the most used Internet protocol.

Web services platform elements:

* SOAP (Simple Object Access Protocol)
* UDDI (Universal Description, Discovery and Integration)
* WSDL (Web Services Description Language)

**Why Web Services?**

Interoperability has Highest Priority

When all major platforms could access the Web using Web browsers, different platforms could interact. For these platforms to work together, Web-applications were developed.

Web-applications are simple applications that run on the web. These are built around the Web browser standards and can be used by any browser on any platform.

Web Services take Web-applications to the Next Level

By using Web services, your application can publish its function or message to the rest of the world.

Web services use XML to code and to decode data, and SOAP to transport it (using open protocols).

With Web services, your accounting department's Win 2k server's billing system can connect with your IT supplier's UNIX server.

Web Services have Two Types of Uses

**Reusable application-components.**

There are things applications need very often. So why make these over and over again?

Web services can offer application-components like: currency conversion, weather reports, or even language translation as services.

**Connect existing software.**

Web services can help to solve the interoperability problem by giving different applications a way to link their data.

With Web services you can exchange data between different applications and different platforms.

What is SOAP?

SOAP is an XML-based protocol to let applications exchange information over HTTP.

Or more simple: SOAP is a protocol for accessing a Web Service.

* SOAP stands for Simple Object Access Protocol
* SOAP is a communication protocol
* SOAP is a format for sending messages
* SOAP is designed to communicate via Internet
* SOAP is platform independent
* SOAP is language independent
* SOAP is based on XML
* SOAP is simple and extensible
* SOAP allows you to get around firewalls
* SOAP is a W3C standard

What is WSDL?

WSDL is an XML-based language for locating and describing Web services.

* WSDL stands for Web Services Description Language
* WSDL is based on XML
* WSDL is used to describe Web services
* WSDL is used to locate Web services
* WSDL is a W3C standard

What is UDDI?

UDDI is a directory service where companies can register and search for Web services.

* UDDI stands for Universal Description, Discovery and Integration
* UDDI is a directory for storing information about web services
* UDDI is a directory of web service interfaces described by WSDL
* UDDI communicates via SOAP
* UDDI is built into the Microsoft .NET platform

## ASP.NET Automates the Process

With ASP.NET, you do not have to write your own WSDL and SOAP documents.

If you look closer at our example Web Service, you will see that ASP.NET has automatically created a WSDL and SOAP request.

**What is the difference between AppDomain, Assembly, Process, and a Thread?**

An AppDomain is a unit of code within a process. AppDomains can be created at runtime, loaded with code, and unloaded. Its an isolation boundary designed to make .NET apps more reliable.

An assembly is a compiled chunk of code. Like an .EXE or a .DLL.

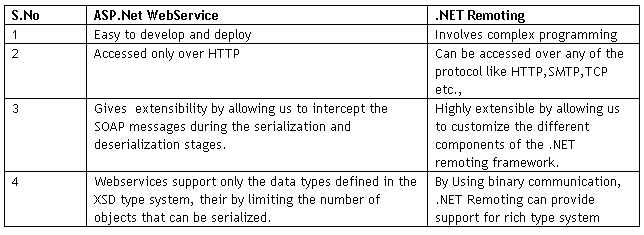
A process is an executing application (waaaay oversimplified).

A thread is an execution context. The operating system executes code within a thread. The operating system switches between threads, allowing each to execute in turn, thus giving the impression that multiple applications are running at the same time.

To put it all together (very simplified)...

A program is executed. A process is created by the operating system, and within its single thread it starts loading code to execute. In a .NET application, a single AppDomain is created by the CLR. The application's executing assembly (the .EXE) is loaded into this AppDomain and begins execution. The application can create AppDomains, load other assemblies into these domains, and then create new Threads to execute code in any of these AppDomains.

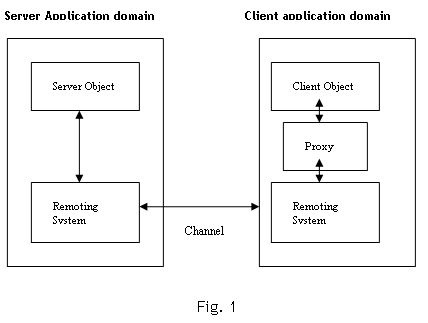
**Compare between .NET Remoting and Web services.**

****

**Architecture:**

Remote objects are accessed through channels. Channels are Transport protocols for passing the messages between Remote objects. A channel is an object that makes communication between a client and a remote object, across app domain boundaries. The .NET Framework implements two default channel classes, as follows:

*HttpChannel:* Implements a channel that uses the HTTP protocol.   
*TcpChannel:* Implements a channel that uses the TCP protocol (Transmission Control Protocol).   
Channel take stream of data and creates package for a transport protocol and sends to other machine. A simple architecture of .NET remoting is as in Fig 1.



As Fig.1 shows, Remoting system creates a proxy for the server object and a reference to the proxy will be returned to the client. When client calls a method, Remoting system sends request thro the channel to the server. Then client receives the response sent by the server process thro the proxy.

**Example:**

Let us see a simple example which demonstrates .Net Remoting. In This example the Remoting object will send us the maximum of the two integer numbers sent.

Creating Remote Server and the Service classes on Machine 1:   
Please note for Remoting support your service (Remote object) should be derived from MarshalByRefObject.

using System;  
using System.Runtime.Remoting.Channels; //To support and handle Channel and channel sinks  
using System.Runtime.Remoting;  
using System.Runtime.Remoting.Channels.Http; //For HTTP channel  
using System.IO;  
namespace ServerApp  
{  
public class RemotingServer  
{  
public RemotingServer()  
{  
//  
// TODO: Add constructor logic here  
//  
}  
}  
//Service class  
public class Service: MarshalByRefObject   
{  
public void WriteMessage (int num1,int num2)   
{  
Console.WriteLine (Math.Max(num1,num2));  
}  
}  
//Server Class  
public class Server  
{  
public static void Main ()   
{  
HttpChannel channel = new HttpChannel(8001); //Create a new channel  
ChannelServices.RegisterChannel (channel); //Register channel  
RemotingConfiguration.RegisterWellKnownServiceType(typeof Service),"Service",WellKnownObjectMode.Singleton);   
Console.WriteLine ("Server ON at port number:8001");  
Console.WriteLine ("Please press enter to stop the server.");  
Console.ReadLine ();  
}  
}  
}

Save the above file as ServerApp.cs. Create an executable by using Visual Studio.Net command prompt by,

csc /r:system.runtime.remoting.dll /r:system.dll ServerApp.cs

A ServerApp.Exe will be generated in the Class folder.

Run the ServerApp.Exe will give below message on the console

Server ON at port number:8001

Please press enter to stop the server.

In order to check whether the HTTP channel is binded to the port, type   
http://localhost:8001/Service?WSDL in the browser.  
You should see a XML file describing the Service class.

Please note before running above URL on the browser your server (ServerApp.Exe should be running) should be ON.

Creating Proxy and the Client application on Machine 2

SoapSuds.exe is a utility which can be used for creating a proxy dll.

Type below command on Visual studio.Net command prompt.

soapsuds -url:http://< Machine Name where service is running>:8001/Service?WSDL -oa:Server.dll

This will generates a proxy dll by name Server.dll. This will be used to access remote object.

**Client Code:**

using System;  
using System.Runtime.Remoting.Channels; //To support and handle Channel and channel sinks  
using System.Runtime.Remoting;  
using System.Runtime.Remoting.Channels.Http; //For HTTP channel  
using System.IO;  
using ServerApp;  
namespace RemotingApp  
{  
public class ClientApp  
{  
public ClientApp()  
{  
}  
public static void Main (string[] args)   
{  
HttpChannel channel = new HttpChannel (8002); //Create a new channel  
ChannelServices.RegisterChannel (channel); //Register the channel  
//Create Service class object  
Service svc = (Service) Activator.GetObject (typeof (Service),"http://<Machine name where Service running>:8001/Service"); //Localhost can be replaced by   
//Pass Message  
svc.WriteMessage (10,20);   
}  
}  
}

Save the above file as ClientApp.cs. Create an executable by using Visual Studio.Net command prompt by,

csc /r:system.runtime.remoting.dll /r:system.dll ClientrApp.cs

A ClientApp.Exe will be generated in the Class folder. Run ClientApp.Exe , we can see the result on Running ServerApp.EXE command prompt.

In the same way we can implement it for TCP channel also.

**WCF**

**What is a service?**

*Services are core business logic that are protocol-independent, location-agnostic and contain no user state.* Services are coarse-grained, meaning the service can perform its logic and return the result in a single call. *Services do not contain presentation logic, so they may be reused across diverse applications.*

#### What is a SOA Service?

SOA is Service Oriented Architecture. SOA service is the encapsulation of a high level business concept. A SOA service is composed of three parts.

* A service class implementing the service to be provided.
* An environment to host the service.
* One or more endpoints to which clients will connect.

#### Explain the important principle of SOA.

*A service-oriented architecture is collection of services which communicate with one another other. The communication can range from simple data passing to two or more services organizing some activity.*

#### What is Windows communication foundation, WCF?

*Windows Communication Foundation (WCF) is an SDK for developing and deploying services on Windows. WCF provides a runtime environment for services, enabling you to expose CLR types as services, and to consume other services as CLR types.*

WCF is part of .NET 3.0 and requires .NET 2.0, so it can only run on systems that support it.

*WCF is a framework that builds applications that can inter-communicate based on service oriented architecture consuming secure and reliable web services. This also helps in Distributed computing.* The WCF also brings together the various communication models into a single model.

WCF is meant for design and deploying distributed applications under [SOA](http://architecture) implementation.

WCF is designed in accordance with [service oriented architecture](http://architecture) principles to support [distributed computing](http://computing) where services are consumed by consumers. Clients can consume multiple services and services can be consumed by multiple clients. Services are [loosely coupled](http://coupling) to each other. Services typically have a [WSDL](http://language) interface (Web Services Description Language) that any WCF client can use to consume the service, irrespective of which platform the service is hosted on. WCF implements many advanced Web services (WS) standards such as [WS-Addressing](http://ws-addressing), [WS-ReliableMessaging](http://ws-reliablemessaging) and [WS-Security](http://ws-security). With the release of .NET Framework 4.0, WCF also provides RSS Syndication Services.

#### What was the code name for WCF?

#### The code name of WCF was Indigo.

#### What are the different technologies involved in WCF?

#### WCF is a unification of .NET framework communication technologies which unites the following technologies:-

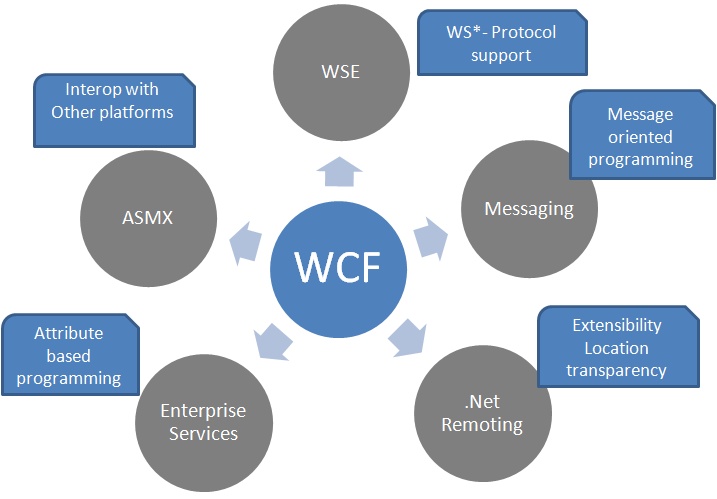
#### NET remoting

#### MSMQ

#### Web services

#### COM+

#### WCF provides a common platform for all .NET communication. Below figures shows the different technology combined to form WCF.



#### What is the advantage of WCF over other technologies?

1. WCF is interoperable with other services when compared to .Net Remoting, where the client and service have to be .Net.
2. WCF services provide better reliability and security in compared to ASMX web services.
3. In WCF, there is no need to make much change in code for implementing the security model and changing the binding. Small changes in the configuration will make your requirements.
4. WCF has integrated logging mechanism, changing the configuration file settings will provide this functionality. In other technology developer has to write the code.

#### Explain the components of WCF - Service class, Hosting environment, END point.

WCF Service is composed of three components:

* **Service class:** It implements the service needed.
* **Host environment:** is an environment that hosts the developed service.
* **Endpoints:** are the connection points for the clients to connect to the service. Clients find the end points through three components like service contract, binding, and address.

#### What is the difference between WCF and Web services?

* Web services can only be invoked by HTTP (traditional webservice with .asmx). While WCF Service or a WCF component can be invoked by any protocol (like http, tcp etc.) and any transport type.
* Second web services are not flexible. However, WCF Services are flexible. If you make a new version of the service then you need to just expose a new end. Therefore, services are agile and which is a very practical approach looking at the current business trends.
* Major Difference is That Web Services Use XmlSerializer But WCF uses DataContractSerializer which is better in Performance as Compared to XmlSerializer.   
  Key issues with XmlSerializer to serialize .NET types to XML are:   
    
  Only Public fields or Properties of .NET types can be translated into XML.

Only the classes which implement IEnumerable interface.

Classes that implement the IDictionary interface, such as Hash table cannot be serialized.   
  
The DataContractAttribute can be applied to the class or a strcture. DataMemberAttribute can be applied to field or a property and theses fields or properties can be either public or private.

* WCF can create services similar in concept to ASMX, but has much more capabilities. WCF is much more efficient than ASP.Net coz it is implemented on pipeline. WCF is more secure, reliable. As WCF is implemented on a different pipeline it does not have all Http capabilities (Http session, cache etc).
* WCF is flexible because its services can be hosted in different types of applications. The following lists several common scenarios for hosting WCF services:
  + IIS
  + WAS
  + Self-hosting
  + Managed Windows Service

We develop WCF as contracts, interface, operations, and data contracts. As the developer we are more focused on the business logic services and need not worry about channel stack. WCF is a unified programming API for any kind of services so we create the service and use configuration information to set up the communication mechanism like HTTP/TCP/MSMQ etc

For more details, read http://msdn.microsoft.com/en-us/library/aa738737.aspx

**What are the differences between WCF and Web services?**

Web service is a part of WCF. WCF offers much more flexibility and portability to develop a service when comparing to web service. Still we are having more advantages over Web service, following table provides detailed difference between them.

| **Features** | **Web Service** | **WCF** |
| --- | --- | --- |
| Hosting | It can be hosted in IIS | It can be hosted in IIS, windows activation service, Self-hosting, Windows service |
| Programming | [WebService] attribute has to be added to the class | [ServiceContract] attribute has to be added to the class |
| Model | [WebMethod] attribute represents the method exposed to client | [OperationContract] attribute represents the method exposed to client |
| Operation | One-way, Request- Response are the different operations supported in web service | One-Way, Request-Response, Duplex are different type of operations supported in WCF |
| XML | System.Xml.serialization name space is used for serialization | System.Runtime.Serialization namespace is used for serialization |
| Encoding | XML 1.0, MTOM(Message Transmission Optimization Mechanism), DIME, Custom | XML 1.0, MTOM, Binary, Custom |
| Transports | Can be accessed through HTTP, TCP, Custom | Can be accessed through HTTP, TCP, Named pipes, MSMQ,P2P, Custom |
| Protocols | Security | Security, Reliable messaging, Transactions |

**Important difference between DataContractSerializer and XMLSerializer.**   
  
\* A practical benefit of the design of the DataContractSerializer is better performance over Xmlserializer.   
\* XML Serialization does not indicate the which fields or properties of the type are serialized into XML where as DataCotratSerializer Explicitly shows the which fields or properties are serialized into XML.   
\* The DataContractSerializer can translate the HashTable into XML.

#### Which namespace is required in a class to use DataContract or DataMember attribute for a class or properties?

**Select from following answers:**

1. **using System.Runtime.Serialization;**
2. using System.ServiceModel;
3. using System.ServiceModel.Web

using System.Runtime.Serialization namespace has all required objects that can be used to make an object serializable. In WCF, in order to expose the object or property as transportable, you need to use DataContract or DataMember

#### Which namespace is used to access WCF service?

**Select from following answers:**

1. System.Service
2. System.Data
3. **System.ServiceModel**
4. System.Web

System.ServiceModel namespace is used to access WCF service.

#### What is endpoint in WCF?

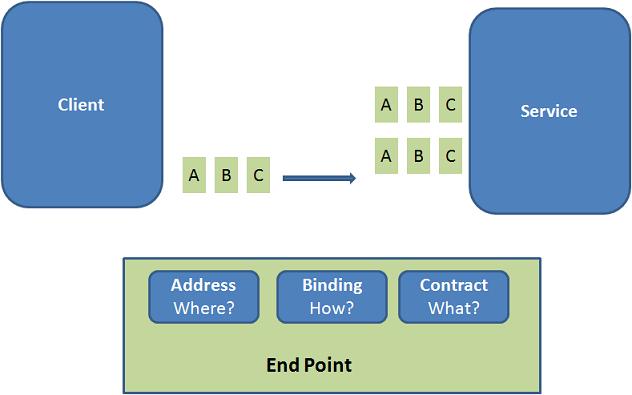
#### Every service must have Address that defines where the service resides, Contract that defines what the service does and a Binding that defines how to communicate with the service. In WCF the relationship between Address, Contract and Binding is called Endpoint.

#### WCF Service is a program that exposes a collection of Endpoints. Each Endpoint is a portal for communicating with the world. All the WCF communications are take place through end point. End point consists of three components. The Endpoint is the fusion of Address, Contract and Binding.

We should remember ABC (where, how and what).

* Address --- Specifies the location of the service which will be like http://Myserver/MyService.Clients will use this location to communicate with our service.
* Binding --- Specifies how the two parties will communicate in term of transport and encoding and protocols
* Contract --- Specifies the interface between client and the server. It’s a simple interface with some attribute.

#### Below figure illustrate the functions of Endpoint



#### Example:

Endpoints will be mentioned in the web.config file on the created service.

<system.serviceModel>

<services>

<service name="MathService"

behaviorConfiguration="MathServiceBehavior">

<endpoint

address="http://localhost:8090/MyService/MathService.svc" contract="IMathService"

binding="wsHttpBinding"/>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="MathServiceBehavior">

<serviceMetadata httpGetEnabled="True"/>

<serviceDebug includeExceptionDetailInFaults="true" />

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

#### What is address in WCF and how many types of transport schemas are there in WCF?

#### Address is a way of letting client know that where a service is located. In WCF, every service is associated with a unique address. This contains the location of the service and transport schemas.

#### WCF supports following transport schemas

#### HTTP

#### TCP

#### Peer network

#### IPC (Inter-Process Communication over named pipes)

#### MSMQ

#### The sample address for above transport schema may look like

#### http://localhost:81

#### http://localhost:81/MyService

#### net.tcp://localhost:82/MyService

#### net.pipe://localhost/MyPipeService

#### net.msmq://localhost/private/MyMsMqService

#### net.msmq://localhost/MyMsMqService

#### What is binding and how many types of bindings are there in WCF?

#### A binding defines how an endpoint communicates to the world. A binding defines the transport (such as HTTP or TCP) and the encoding being used (such as text or binary). A binding can contain binding elements that specify details like the security mechanisms used to secure messages, or the message pattern used by an endpoint.

#### WCF supports nine types of bindings.

| **Binding** | **Description** |
| --- | --- |
| BasicHttpBinding | Basic Web service communication. No security by default |
| WSHttpBinding | Web services with WS-\* support. Supports transactions |
| WSDualHttpBinding | Web services with duplex contract and transaction support |
| WSFederationHttpBinding | Web services with federated security. Supports transactions |
| MsmqIntegrationBinding | Communication directly with MSMQ applications. Supports transactions |
| NetMsmqBinding | Communication between WCF applications by using queuing. Supports transactions |
| NetNamedPipeBinding | Communication between WCF applications on same computer. Supports duplex contracts and transactions |
| NetPeerTcpBinding | Communication between computers across peer-to-peer services. Supports duplex contracts |
| NetTcpBinding | Communication between WCF applications across computers. Supports duplex contracts and transactions |

#### What are contracts in WCF?

#### In WCF, all services expose contracts. The contract is a platform-neutral and standard way of describing what the service does. WCF defines four types of contracts:

#### Service contracts,

#### Data contracts,

#### Fault contracts and

#### Message contracts.

#### Service contracts

#### Describe which operations the client can perform on the service.

#### There are two types of Service Contracts.

#### ServiceContract - This attribute is used to define the Interface.

#### OperationContract - This attribute is used to define the method inside Interface.

To create a service contract you define an interface with related methods representative of a collection of service operations, and then decorate the interface with the *ServiceContract* Attribute to indicate it is a service contract. Methods in the interface that should be included in the service contract are decorated with the *OperationContract* Attribute.

[ServiceContract()]

public interface ISimpleCalculator

{

[OperationContract()]

int Add(int num1, int num2);

}

Once we define Service contract in the interface, we can create implement class for this interface.

public class SimpleCalculator : ISimpleCalculator

{

public int Add(int num1, int num2)

{

return num1 + num2;

}

}

#### Data contracts

#### Define which data types are passed to and from the service. WCF defines implicit contracts for built-in types such as int and string, but we can easily define explicit opt-in data contracts for custom types.

#### There are two types of Data Contracts.

#### DataContract - attribute used to define the class

#### DataMember - attribute used to define the properties.

We need to include *System.Runtime.Serialization* reference to the project. This assembly holds the *DataContract* and *DataMember* attribute.

Create user defined data type called Employee. This data type should be identified for serialization and deserialization by mentioning with [DataContract] and [DataMember] attribute.

[ServiceContract]

public interface IEmployeeService

{

[OperationContract]

Employee GetEmployeeDetails(int EmpId);

}

[DataContract]

public class Employee

{

private string m\_Name;

private int m\_Age;

private int m\_Salary;

private string m\_Designation;

private string m\_Manager;

[DataMember]

public string Name

{

get { return m\_Name; }

set { m\_Name = value; }

}

[DataMember]

public int Age

{

get { return m\_Age; }

set { m\_Age = value; }

}

[DataMember]

public int Salary

{

get { return m\_Salary; }

set { m\_Salary = value; }

}

[DataMember]

public string Designation

{

get { return m\_Designation; }

set { m\_Designation = value; }

}

[DataMember]

public string Manager

{

get { return m\_Manager; }

set { m\_Manager = value; }

}

}

Implementation of the service class is shown below. In GetEmployee method we have created the Employee instance and return to the client. Since we have created the data contract for the Employee class, client will aware of this instance whenever he creates proxy for the service.

public class EmployeeService : IEmployeeService

{

public Employee GetEmployeeDetails(int empId)

{

Employee empDetail = new Employee();

//Do something to get employee details and assign to 'empDetail' properties

return empDetail;

}

}

### Client side

On client side we can create the proxy for the service and make use of it. The client side code is shown below.

protected void btnGetDetails\_Click(object sender, EventArgs e)

{

EmployeeServiceClient objEmployeeClient = new EmployeeServiceClient();

Employee empDetails;

empDetails = objEmployeeClient.GetEmployeeDetails(empId);

//Do something on employee details

}

#### If DataMember attributes are not specified for a properties in the class that property can't be passed to-from web service.

#### Fault contracts

#### Define which errors are raised by the service, and how the service handles and propagates errors to its clients.

Service that we develop might get error in come case. This error should be reported to the client in proper manner. Basically when we develop managed application or service, we will handle the exception using try- catch block. But these exceptions handlings are technology specific.

In order to support interoperability and client will also be interested only, what wents wrong? not on how and where cause the error.

By default when we throw any exception from service, it will not reach the client side. WCF provides the option to handle and convey the error message to client from service using SOAP Fault contract.

Suppose the service I consumed is not working in the client application. I want to know the real cause of the problem. How I can know the error? For this we are having Fault Contract. Fault Contract provides documented view for error accorded in the service to client. This help as to easy identity the what error has accord. Let us try to understand the concept using sample example.

**Step 1:** I have created simple calculator service with Add operation which will throw general exception as shown below

//Service interface

[ServiceContract()]

public interface ISimpleCalculator

{

[OperationContract()]

int Add(int num1, int num2);

}

//Service implementation

public class SimpleCalculator : ISimpleCalculator

{

public int Add(int num1, int num2)

{

//Do something

throw new Exception("Error while adding number");

}

}

**Step 2:** On client side code. Exceptions are handled using try-Catch block. Even though I have capture the exception when I run the application. I got the message that exceptions are not handled properly.

try

{

MyCalculatorServiceProxy.MyCalculatorServiceProxy proxy

= new MyCalculatorServiceProxy.MyCalculatorServiceProxy();

Console.WriteLine("Client is running at " + DateTime.Now.ToString());

Console.WriteLine("Sum of two numbers... 5+5 =" + proxy.Add(5, 5));

Console.ReadLine();

}

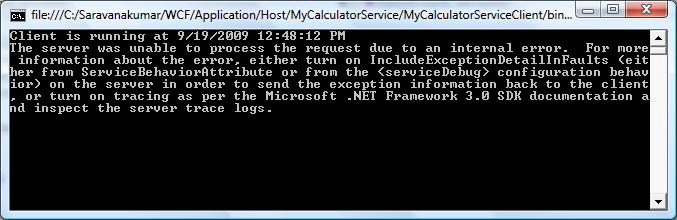
catch (Exception ex)

{

Console.WriteLine(ex.Message);

Console.ReadLine();

}



**Step 3:** Now if you want to send exception information form service to client, you have to use FaultException as shown below.

public int Add(int num1, int num2)

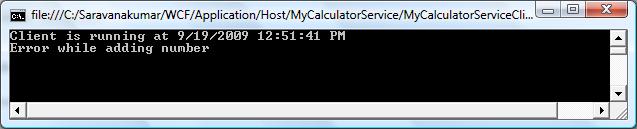
{

//Do something

throw new FaultException("Error while adding number");

}

**Step 4:** Output window on the client side is show below.



**Step 5:** You can also create your own Custom type and send the error information to the client using *FaultContract*. These are the steps to be followed to create the fault contract.

* Define a type using the data contract and specify the fields you want to return.
* Decorate the service operation with the FaultContract attribute and specify the type name.
* Raise the exception from the service by creating an instance and assigning properties of the custom exception.

**Step 6:** Defining the type using Data Contract

[DataContract()]

public class CustomException

{

[DataMember()]

public string Title;

[DataMember()]

public string ExceptionMessage;

[DataMember()]

public string InnerException;

[DataMember()]

public string StackTrace;

}

**Step 7:** Decorate the service operation with the *FaultContract*

[ServiceContract()]

public interface ISimpleCalculator

{

[OperationContract()]

[FaultContract(typeof(CustomException))]

int Add(int num1, int num2);

}

**Step 8:** Raise the exception from the service

public int Add(int num1, int num2)

{

//Do something

CustomException ex = new CustomException();

ex.Title = "Error Funtion:Add()";

ex.ExceptionMessage = "Error occur while doing add function.";

ex.InnerException = "Inner exception message from serice";

ex.StackTrace = "Stack Trace message from service.";

throw new FaultException(ex,"Reason: Testing the Fault contract") ;

}

**Step 9:** On client side, you can capture the service exception and process the information, as shown below.

try

{

MyCalculatorServiceProxy.MyCalculatorServiceProxy proxy

= new MyCalculatorServiceProxy.MyCalculatorServiceProxy();

Console.WriteLine("Client is running at " + DateTime.Now.ToString());

Console.WriteLine("Sum of two numbers... 5+5 =" + proxy.Add(5, 5));

Console.ReadLine();

}

catch (FaultException<MyCalculatorService.CustomException> ex)

{

//Process the Exception

}

#### Message contracts

#### Allow the service to interact directly with messages. Message contracts can be typed or un-typed, and are useful in interoperability cases and when there is an existing message format we have to comply with.

Message is the packet of data which contains important information. WCF uses these messages to transfer information from Source to destination.

WCF uses SOAP(Simple Object Access Protocol) Message format for communication. SOAP message contain Envelope, Header and Body.SOAP envelope contails name, namespace,header and body element. SOAP Hear contain important information which are not directly related to message. SOAP body contains information which is used by the target.

Diagram Soap envelope

### Message Pattern

It describes how the programs will exchange message each other. There are three way of communication between source and destination

1. **Simplex** - It is one way communication. Source will send message to target, but target will not respond to the message.
2. **Request/Replay** - It is two way communications, when source send message to the target, it will resend response message to the source. But at a time only one can send a message
3. **Duplex** - It is two way communication, both source and target can send and receive message simultaniouly.

### What is Message contract?

As I said earlier, WCF uses SOAP message for communication. Most of the time developer will concentrate more on developing the DataContract, Serializing the data, etc. WCF will automatically take care of message. On Some critical issue, developer will also require control over the SOAP message format. In that case WCF provides Message Contract to customize the message as per requirement.

WCF supports either RPC(Remote Procedure Call) or Message style operation model. In the RPC model, you can develop operation with Ref and out parameter. WCF will automatically create the message for operation at run time. In Message style operation WCF allows to customize the message header and define the security for header and body of the message.

### Defining Message Contract

Message contract can be applied to type using MessageContract attribute. Custom Header and Body can be included to message using 'MessageHeader' and '*MessageBodyMember*'atttribute. Let us see the sample message contract definition.

[MessageContract]

public class EmployeeDetails

{

[MessageHeader]

public string EmpID;

[MessageBodyMember]

public string Name;

[MessageBodyMember]

public string Designation;

[MessageBodyMember]

public int Salary;

[MessageBodyMember]

public string Location;

}

When I use this EmployeeDeatils type in the service operation as parameter. WCF will add extra header call 'EmpID' to the SOAP envelope. It also add Name, Designation, Salary, Location as extra member to the SOAP Body.

### Rules :

You have to follow certain rules while working with Message contract

1. When using Message contract type as parameter, Only one parameter can be used in servicie Operation

[OperationContract]

void SaveEmployeeDetails(EmployeeDetails emp);

1. Service operation either should return Messagecontract type or it should not return any value

[OperationContract]

EmployeeDetails GetEmployeeDetails();

1. Service operation will accept and return only message contract type. Other data types are not allowed.

[OperationContract]

EmployeeDetails ModifyEmployeeDetails(EmployeeDetails emp);

**Note:** If a type has both Message and Data contract, service operation will accept only message contract.

# MessageHeaderArray Attribute

Consider the Message contract type definition as shown below.

[MessageContract]

public class Department

{

[MessageHeader]

public string DepartmentID;

[MessageHeader]

public string DepartmentName;

[MessageHeader]

public Employees Employee();

}

In this we are having array of Employee type as message header. When this converted to SOAP Header it looks as shown below.

<Department>

<DepartmentID>PRO1243</DepartmentID>

<DepartmentName>Production</DepartmentName>

<Employees>

<Employee>Sam</Employee>

<Employee>Ram</Employee>

<Employee>Raja</Employee>

</Employees>

</Department>

Suppose you want to show the all employee detail in same level. We can use MessageHeaderArray attribute which will serialize the array element independently. If you use the MessageHeaderArray attribute of Employees, SOAP message will look as shown below.

<Department>

<DepartmentID>PRO1243</DepartmentID>

<DepartmentName>Production</DepartmentName>

<Employee>Sam</Employee>

<Employee>Ram</Employee>

<Employee>Raja</Employee>

</Department>

**Note:** MessageHeaderArray Attribute is applicable only for Array, not for collection.

# Message Contract Properties

### ProtectionLevel

You can mention the *MessageHeader* or *MessageBodyMember* to be signed or Encrypted using *ProtectionLevel* property.

**Example**

using System.Net.Security;

[MessageContract]

public class EmployeeDetails

{

[MessageHeader(ProtectionLevel=ProtectionLevel.None)]

public string EmpID;

[MessageBodyMember(ProtectionLevel = ProtectionLevel.Sign )]

public string Name;

[MessageBodyMember(ProtectionLevel = ProtectionLevel.Sign )]

public string Designation;

[MessageBodyMember(ProtectionLevel=ProtectionLevel.EncryptAndSign)]

public int Salary;

}

In the above type definition, we have made the different protection level for body. But the protection level of the body is determind by the highest *ProtectionLevel* property. By default if you are not specifying the protection level it takes 'EncryptAndSign'. So it good if you specify minimum ProtectionLevel required.

### Name and Namespace:

SOAP representation of the message element can be change by mentioning Name and Namespace property of the Header and Body member. By default namespace is the same as the namespace of the service contract that the message is participating. In the below example, I have mention the Name property to the EmpID and Name.

[MessageContract]

public class EmployeeDetails

{

[MessageHeader(Name="ID")]

public string EmpID;

[MessageBodyMember(Name="EmployeeName")]

public string Name;

[MessageBodyMember()]

public string Designation;

[MessageBodyMember()]

public int Salary;

}

When SOAP message representation, its name is changed to ID and EmployeeName.

<EmployeeDetails>

<ID>45634</ID>

<EmployeeName>Sam</EmployeeName>

<Designation>Software Engineer</Designation>

<Salary>25000</Salary>

</EmployeeDetails>

### Order

The order of the body elements are alpehabetical by default. But you can control the order, usiing *Order* property in the *MessageBody* attribute.

[MessageContract]

public class EmployeeDetails

{

[MessageHeader()]

public string EmpID;

[MessageBodyMember(Order=2)]

public string Name;

[MessageBodyMember(Order=3)]

public string Designation;

[MessageBodyMember(Order=1)]

public int Salary;

}

#### What are various ways of hosting WCF Services?

There are three major ways of hosting a WCF services

* Self-hosting the service in his own application domain. This we have already covered in the first section. The service comes in to existence when you create the object of Service Host class and the service closes when you call the Close of the Service Host class.
* Host in application domain or process provided by IIS Server.
* Host in Application domain and process provided by WAS (Windows Activation Service) Server.

More details http://www.dotnetfunda.com/articles/article221.aspx#whatarethevariouswaysofhostingaWCFservice

#### What is the use of ServiceBehavior attribute in WCF? Or How instance management is done in WCF?

ServiceBehaviour attribute is used to specify the InstanceContextMode for the WCF Service class (This can be used to maintained a state of the service or a client too)

There are three instance modes in the WCF:

* Per-Session: This is used to create a new instance for a service and the same instance is used for all method for a particular client. (eg: State can be maintained per session by declaring a variable)
* Per-Call: This is used to create a new instance for every call from the client whether same client or different. (eg: No state can be maintained as every time a new instance of the service is created)
* Single/Singleton: This is used to create only one instance of the service and the same instance is used for the entire client request. (eg: Global state can be maintained but this will be applicable for all clients)

#### 

### Configuration:

Instance mode can be configured using *ServiceBehavior* attribute. This can be specified at implementing the service contract as shown below.

[ServiceContract()]

public interface IMyService

{

[OperationContract]

int MyMethod();

}

[ServiceBehavior(InstanceContextMode=InstanceContextMode.Single)]

public class MyService:IMyService

{

public int MyMethod()

{

//Do something

}

}

#### What is service and client in perspective of data communication?

#### A service is a unit of functionality exposed to the world.

#### The client of a service is merely the party consuming the service.

#### What is duplex contract in WCF?

Duplex contract: It enables clients and servers to communicate with each other. The calls can be initiated independently of the other one. The duplex contract is one the message patterns available to Windows Communication Foundation (WCF) services. It comprises of two one-way contracts.

#### Different transaction isolation levels in WCF

* Read Uncommitted: - Also known as Dirty isolation level. It makes sure that corrupt Data cannot be read. This is the lowest isolation level
* Read Committed: - It ensures not to read the data that has been changed by any other application and is not yet committed. It is the default level.
* Repeatable Read: - It stops the usage of dirt read and non-repeatable read. It states that data fetched through a query will be locked and will not be updated by any other transaction.
* Serializable: - It does not allow any modification and addition of new data till the transaction is completed. This is considered to be a very restrictive level.
* Snapshot: - It raises error on modifying a data that has already been changed by any transaction.

#### How to set the timeout property for the WCF Service client call?

#### The timeout property can be set for the WCF Service client call using binding tag.

#### <client>

#### <endpoint

#### ...

#### binding = "wsHttpBinding"

#### bindingConfiguration = "LongTimeout"

#### ...

#### />

#### </client>

#### <bindings>

#### <wsHttpBinding>

#### <binding name = "LongTimeout" sendTimeout = "00:04:00"/>

#### </wsHttpBinding>

#### </bindings>

#### If no timeout has been specified, the default is considered as 1 minute.

#### How to deal with operation overloading while exposing the WCF services?

#### By default overload operations (methods) are not supported in WSDL based operation. However by using Name property of OperationContract attribute, we can deal with operation overloading scenario.

#### [ServiceContract]

#### interface ICalculator

#### {

#### [OperationContract(Name = "AddInt")]

#### int Add(int arg1,int arg2);

#### [OperationContract(Name = "AddDouble")]

#### double Add(double arg1,double arg2);

#### }

#### Notice that both method name in the above interface is same (Add), however the Name property of the OperationContract is different. In this case client proxy will have two methods with different name AddInt and AddDouble.

#### How to configure Reliability while communicating with WCF Services?

#### Reliability can be configured in the client config file by adding reliableSession under binding tag.

#### <system.serviceModel>

#### <services>

#### <service name = "MyService">

#### <endpoint

#### address = "net.tcp://localhost:8888/MyService"

#### binding = "netTcpBinding"

#### bindingConfiguration = "ReliableCommunication"

#### contract = "IMyContract"

#### />

#### </service>

#### </services>

#### <bindings>

#### <netTcpBinding>

#### <binding name = "ReliableCommunication">

#### <reliableSession enabled = "true"/>

#### </binding>

#### </netTcpBinding>

#### </bindings>

#### </system.serviceModel>

#### Reliability is supported by following bindings only

#### NetTcpBinding

#### WSHttpBinding

#### WSFederationHttpBinding

#### WSDualHttpBinding

#### What is Transport and Message Reliability?

#### Transport reliability (such as the one offered by TCP) offers point-to-point guaranteed delivery at the network packet level, as well as guarantees the order of the packets. Transport reliability is not resilient to dropping network connections and a variety of other communication problems.

#### Message reliability deals with reliability at the message level independent of how many packets are required to deliver the message. Message reliability provides for end-to-end guaranteed delivery and order of messages, regardless of how many intermediaries are involved, and how many network hops are required to deliver the message from the client to the service.

#### What are different elements of WCF Srevices Client configuration file?

#### WCF Services client configuration file contains endpoint, address, binding and contract. A sample client config file looks like

#### <system.serviceModel>

#### <client>

#### <endpoint name = "MyEndpoint"

#### address = "http://localhost:8000/MyService/"

#### binding = "wsHttpBinding"

#### contract = "IMyContract"

#### />

#### </client>

#### </system.serviceModel>

#### What is Proxy and how to generate proxy for WCF Services?

#### The proxy is a CLR class that exposes a single CLR interface representing the service contract. The proxy provides the same operations as service's contract, but also has additional methods for managing the proxy life cycle and the connection to the service. The proxy completely encapsulates every aspect of the service: its location, its implementation technology and runtime platform, and the communication transport.

#### The proxy can be generated using Visual Studio by right clicking Reference and clicking on Add Service Reference. This brings up the Add Service Reference dialog box, where you need to supply the base address of the service (or a base address and a MEX URI) and the namespace to contain the proxy.

#### Proxy can also be generated by using SvcUtil.exe command-line utility. We need to provide SvcUtil with the HTTP-GET address or the metadata exchange endpoint address and, optionally, with a proxy filename. The default proxy filename is output.cs but you can also use the /out switch to indicate a different name.

#### SvcUtil http://localhost/MyService/MyService.svc /out:Proxy.cs

#### When we are hosting in IIS and selecting a port other than port 80 (such as port 88), we must provide that port number as part of the base address:

#### SvcUtil http://localhost:88/MyService/MyService.svc /out:Proxy.cs

#### What is the address formats of the WCF transport schemas?

#### Address format of WCF transport schema always follow

#### [transport]://[machine or domain][:optional port] format.

#### for example:

#### HTTP Address Format

#### http://localhost:8888

#### the way to read the above url is

#### "Using HTTP, go to the machine called localhost, where on port 8888 someone is waiting"

#### When the port number is not specified, the default port is 80.

#### TCP Address Format

#### net.tcp://localhost:8888/MyService

#### When a port number is not specified, the default port is 808:

#### net.tcp://localhost/MyService

#### NOTE: Two HTTP and TCP addresses from the same host can share a port, even on the same machine.

#### IPC Address Format

#### net.pipe://localhost/MyPipe

#### We can only open a named pipe once per machine, and therefore it is not possible for two named pipe addresses to share a pipe name on the same machine.

#### MSMQ Address Format

#### net.msmq://localhost/private/MyService

#### net.msmq://localhost/MyService

#### How to define a service as REST based service in WCF?

#### WCF 3.5 provides explicit support for RESTful communication using a new binding named WebHttpBinding.

#### The below code shows how to expose a RESTful service

#### [ServiceContract]

#### interface IStock

#### {

#### [OperationContract]

#### [WebGet]

#### int GetStock(string StockId);

#### }

#### By adding the WebGetAttribute, we can define a service as REST based service that can be accessible using HTTP GET operation.