Mahindra Satyam

Workbook for WCF

A beginner's guide to effective programming

(In C#.Net)

Why This Module

The move to service-oriented communication has changed software development. Viewing services as a distinct software abstraction is fundamental to service-oriented architecture (SOA), an approach that many organizations are putting in place today. Whether implemented using SOAP or in some other way, applications that interact through services are becoming the norm.

Software development environments must keep pace with these changes. The benefits services bring should be reflected in the tools and technologies that developers use. Windows Communication foundation (WCF), Microsoft's technology for service-oriented applications, is designed to address these requirements.

First released as part of the .NET Framework 3.0 in 2006, an updated version of this technology is included in the .NET Framework 3.5. For a large share of new software built on .NET, WCF is the right foundation.

Windows Communication Foundation (WCF) is Microsoft's unified programming model for building service-oriented applications. It enables developers to build secure, reliable, transacted solutions that integrate across platforms and interoperate with existing investments.

In this module, participants will be introduced to WCF. They will learn about WCF, its advantages and disadvantages and why there is need for WCF. Students will then learn to create a WCF service and to host and consume this WCF service.

Guide to Use this Workbook

Conventions Used

Convention	Description
Topic	Indicates the Topic Statement being discussed.
Estimated Time	Gives an idea of estimated time needed to understand the Topic and complete the Practice session.
Presentation	Gives a brief introduction about the Topic.
Scenario	Gives a real time situation in which the Topic is used.
Demonstration/Code Snippet	Gives an implementation of the Topic along with Screenshots and real time code.
`Code in Italic`	Represents a few lines of code in Italics which is generated by the System related to that particular event.
// OR '	Represents a few lines of code (Code Snippet) from the complete program which describes the Topic.
Context	Explains when this Topic can be used in a particular Application.
Practice Session	Gives a practice example for the participant to implement the Topic, which gives him a brief idea how to develop an Application using the Topic.
Check list	Lists the brief contents of the Topic.
Common Errors	Lists the common errors that occur while developing the Application.
Exceptions	Lists the exceptions which result from the execution of the Application.

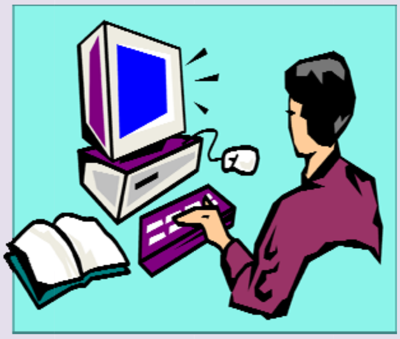
Lessons Learnt	Lists the lessons learnt from the article of the workbook.
Best Practices	Lists the best ways for the efficient development of the Application.
Notes	Gives important information related to the Topic in the form of a note

Contents

1.0 Introduction to WCF

Topics

- Distributed Computing APIs
- Role of DCOM
- Merits and demerits of DCOM
- Role of COM+
- Merits and demerits of COM+
- Role of .Net Remoting
- Merits and demerits of Remoting
- Role of XML Web Services
- Merits and demerits of XML Web Services
- Overview of SOA
- Introduction to WCF and its Architecture
- WCF Features



Topic: Distributed Computing APIs

Estimated Time: 5 minutes.

Objectives: This module will familiarize the participant with

- The importance of distributed computing APIs?
- Different distributed Computing APIs

Presentation:

There are a number of existing approaches to building distributed applications. The Windows operating system has historically provided a number of APIs for building distributed systems. While it is true that most people consider a "distributed system" to involve at the very least two networked computers, this term in the broader sense can simply refer to two executables that need to exchange data, even if they happen to be running on the same physical machine. Based on this description, selecting a required distributed API for an application, one must know, **will this application be used internally or will external users require access to the application functionality?** If we have to build a distributed system for internal use, we have a far greater chance of ensuring that each connected computer is running the same operating system, using the same programming framework (.NET, COM, J2EE etc) and will be able to leverage existing security system for purposes of authentication, authorization, and so forth.

In contrast, to build a system that must be reached by external users, we have a whole other set of issues to contend with. We will most likely not be able to dictate to external users which operating system they make use of, which programming framework they use to build their software, or how they configure their security settings. In larger company or in a university setting that makes use of numerous operating systems and programming technologies, an internal application suddenly faces the same challenges as an application accessed by external users. In either of these cases, there is need to limit ourselves to a more flexible distributed API to ensure the furthest "reach" of application.

Based on the information to this distributed computing question, next step is to pinpoint exactly which API (or a collection of APIs) to be used in application.

As we all know different distributed computing APIs are available, which include Web services, .NET Remoting, Message Queuing (MSMQ) and DCOM/COM+ etc. Quick recap of some of the major distributed APIs historically used by Windows software developers will easily be able to see the usefulness of Windows Communication Foundation.

Topic: Role of DCOM

Estimated Time: 10 minutes.

Objectives: This module will familiarize the participant with

- Role of DCOM in distributed computing
- The main features of DCOM.

Presentation:

- Microsoft developed COM to enable applications to interact with each other and to promote reusability.
- COM is the set of specifications that, when followed, allows software components to communicate with each other.
- Each component exposes its functionality through an interface and is uniquely identified by global unique identifiers (GUIDs).
- The advantage of using COM is that different components developed in different languages can write these software components and interact with each other by using IUnknown and other standard COM interfaces.
- Most of Microsoft's products, including Microsoft Office, SQL Server, and even Windows, are based on COM. Though COM provides the ability to reuse the components locally, it was not designed to work well with remote components.
- Few specifications and extensions had been made that were based on COM and that interacted with remote components. However, the need for remote method invocations grew substantially. To solve this concern, Microsoft developed DCOM. This essentially is a combination of COM and the network protocol that allows you to run a COM object on a remote computer.
- DCOM was a proprietary wire-protocol standard from Microsoft to extend COM so it could work in distributed environments.
- DCOM provides an opportunity to distribute your component across different locations according to the application requirements.
- In addition, DCOM provides basic infrastructure support such as reliability, security, location independence, and efficient communication between COM objects that are residing across processes and machines.

MERITS OF DCOM

1) Location Independence:

- Some components can only be run on specific machines or at specific locations.

- Smaller components increase flexibility of deployment, but they also increase network traffic.
- Larger components reduce network traffic, but they also reduce flexibility of deployment.
- With DCOM these critical design constraints are fairly easy to work around, because the details of deployment are not specified in the source code.
- DCOM completely hides the location of a component, whether it is in the same process as the client or on a machine halfway around the world.
- In all cases, the way the client connects to a component and calls the component's methods is identical.
- Not only does DCOM require no changes to the source code, it does not even require that the program be recompiled. A simple reconfiguration changes the way components connect to each other.

2) Language Neutrality:

- Language independence also enables rapid prototyping: components can be first developed in a higher-level language, such as Microsoft Visual Basic, and later re implemented in a different language, such as C++ or Java, that can better takes advantage of advanced features such as DCOM's free threading, free multithreading and thread pooling.

3) Scalability:

- A critical factor for a distributed application is its ability to grow with the number of users, the amount of data, and the required functionality.
- The application should be small and fast when the demands are minimal, but it should be able to handle additional demands without sacrificing performance or reliability. DCOM provides a number of features that enhance your application's scalability.

4) Flexible Deployment:

- DCOM's location independence makes it easy to distribute components over other computers, offering an easier and less expensive route to scalability.

DEMERITS OF DCOM

DCOM and other distributed technologies such as CORBA, RMI, and so on, are based on several assumptions. One of the key assumptions is that one organization will manage all the components in the systems that are interacting with each other. Another is that the location of a component will not vary from one place to the other. This scenario can work fine within an organization, but as you cross organization boundaries, the limitations of DCOM become more significant.

- Microsoft has invested a lot in DCOM to ensure that calling a remote method is as simple as calling the local component by simplifying the low-level network communication requirements. Most of the time this resulted in bad programming practices by programmers, which resulted in increased network traffic and performance bottlenecks.
- DCOM, is based on a proprietary standard and essentially built taking only the Windows operating systems into account, making it unsuitable for heterogeneous environments.

- Another issue with DCOM is that its client is tightly coupled with the server, so any changes done on the client mandates a modification on the server.
- System administrators need to compromise the security of the firewall in order to use DCOM across firewalls/locations.
- DCOM is used to communicate through ports that are generally restricted by firewalls because the ports are susceptible to attacks

Context:

- To understand the importance of using DCOM
- Understanding the merits and demerits of DCOM

Lessons Learnt:

- DCOM provides an opportunity to distribute your component across different locations according to the application requirements.
- A combination of COM and the network protocol that allows you to run a COM object on a remote computer.
- COM's language independence, application developers can choose the tools and languages that they are most familiar with.
- DCOM's location independence makes it easy to distribute components over other computers, offering an easier and less expensive route to scalability.
- DCOM came before the computer world experienced the Internet boom

Best Practices:

- It is recommended practice to use DCOM with the systems supporting windows operating system

Topic: Role of COM+

Estimated Time: 10 minutes.

Objectives: This module will familiarize the participant with

- Role of COM+ in distributed computing
- Understanding the main features of COM+
- Merits and Demerits of COM+

Presentation:

- COM+ is the next step in the evolution of the Microsoft® Component Object Model and Microsoft Transaction Server (MTS).
- COM+ handles many of the resource management tasks you previously had to program yourself, such as thread allocation and security.
- It automatically makes your applications more scalable by providing thread pooling, object pooling, and just-in-time object activation.
- COM+ also helps protect the integrity of your data by providing transaction support, even if a transaction spans multiple databases over a network.
- If you are a system administrator, you will be installing, deploying, and configuring COM+ applications and their components.
- If you are an application programmer, you will be writing components and integrating them as applications.
- If you are a tools vendor, you will be developing or modifying tools to work in the COM+ environment.
- COM+ version 1.5 adds new features that are designed to increase the overall scalability, availability, and manageability of COM+ applications both for developers and for system administrators.

Merits of COM+

- An advantage of COM+ was that it could be run in "component farms". A component, if coded properly, could be reused by new calls to its initializing routine without unloading it from memory. Components could also be distributed (called from another machine) as was previously only possible with DCOM.
- COM+ also introduced a subscriber/publisher event mechanism called COM+ Events, and provided a new way of leveraging MSMQ (inter-application asynchronous messaging) with components called Queued Components.
- COM+ events extend the COM+ programming model to support late-bound events or method calls between the publisher or subscriber and the event system.

Demerits of COM+

- Have to have strong named assemblies - not optional
- Can't use any non-strong named assemblies in your project
- Easy wind up with some sort of configuration/versioning problem

Lessons Learnt:

☑ COM+ handles many of the resource management tasks you previously had to program yourself

☑ COM+ also helps protect the integrity of your data by providing transaction support, even if a transaction spans multiple databases over a network.

☑ COM+ helps protect the integrity of your data by providing transaction support, even if a transaction spans multiple databases over a network.

☑ COM+ events extend the COM+ programming model to support late-bound events or method calls between the publisher or subscriber and the event system.

Topic: Role of .Net Remoting

Estimated Time: 10 minutes.

Objectives: This module will familiarize the participant with

- Role of .Net Remoting in Distributed Computing.
- Understanding the features of .Net Remoting.
- Merits and Demerits of .Net Remoting

Presentation:

- Though COM and DCOM are able to provide reusability and a distributed platform, they also suffer from problems of versioning, reference counting, and so on.
- Microsoft .NET came up with a vision to be more connected than ever. It wanted to deliver software as a “service” and also resolve issues related to COM. The release of .NET was termed as the biggest revolution ever on the Microsoft platform after the introduction of Windows.
- .NET Remoting is one of the ways to create distributed applications in .NET. Developers now have additional options such as XML web services and service components.
- It replaces DCOM as the preferred technology for building distributed applications. It addresses problems that have wounded distributed applications for many years (that is, interoperability support, extensibility support, efficient lifetime management, custom hosts, and an easy configuration process).
- .NET Remoting delivers on the promises of easy distributed computing by providing a simple, extensible programming model, without compromising flexibility, scalability, and robustness.
- It comes with a default implementation of components such as channels and protocols, but all of them are pluggable and can be replaced with better options without much code modification. Earlier, processes were used to isolate applications from each other. Each process had its own virtual address space, and the code that ran in one process could not access the code or data of another process.

MERITS OF REMOTING

- .NET Remoting delivers on the promises of easy distributed computing by providing a simple, extensible programming model, without compromising flexibility, scalability, and robustness.
- Not only does .NET Remoting enable communication between application domains, but it also can be extended across processes, machines, and networks.
- It is flexible in the channels and formatters that can be used and has a wide variety of options to maintain state.

DEMERITS OF REMOTING

- .NET Remoting works best when assemblies that define the types that are used to integrate are shared. .NET Remoting works fairly well if there is full control over both end of the wire. Therefore, it works well in an intranet where you have complete control of the deployment, the versioning, and the testing.

- .NET Remoting is proprietary to .NET and works seamlessly to Exchange data between two .NET applications. It is deeply rooted in the common language runtime (CLR) and relies on the CLR to obtain metadata. This metadata means the client must understand .NET in order to communicate with endpoints exposed by .NET Remoting.
- .NET Remoting requires a big leap between programming at a high level and Dropping down into the infrastructure.
- It’s pretty easy to code .NET Remoting with the available components, but if you want to start learning about adding your own transports, the level of complexity increases.
- .NET Remoting suffers from the issues of load balancing because it is not intelligent enough to shift a request from a busy application server to one that is not as busy.

Context:

- Understanding the .Net Remoting
- Advantages over COM and DCOM
- Understanding the Merits and Demerits of .Net Remoting

Lessons Learnt:

☑ .NET Remoting delivers on the promises of easy distributed computing by providing a simple, extensible programming model, without compromising flexibility, scalability, and robustness.

☑ .NET Remoting is one of the ways to create distributed applications in .NET. Developers now have additional options such as XML web services and service components.

☑ NET Remoting works best when assemblies that define the types that are used to integrate are shared.

☑ .NET Remoting works fairly well if there is full control over both end of the wire. Therefore, it works well in an intranet where you have complete control of the deployment, the versioning, and the testing.

☑ .NET Remoting delivers on the promises of easy distributed computing by providing a simple, extensible programming model, without compromising flexibility, scalability and robustness.

Topic: Role of XML Web Services

Estimated Time: 10 minutes.

Objectives: This module will familiarize the participant with

- Role of XML Web Services in Distributed Computing.
- Understanding the features of XML Web Services.
- Merits and Demerits of XML Web Services.

Presentation:

- A web service is a component on the Web that is accessible through open standards such as SOAP and HTTP.
- Accessing a remote component is not a new concept. It was previously accomplished through RMI, CORBA, and DCOM.
- But these distributed components are based on proprietary standards or protocols.
- Earlier distributed technologies had two main problems:
    - Interoperability
    - Crossing firewalls
- These proprietary standards cannot interoperate with each other because they have their own binary standards and protocols.
- Additionally, they send binary messages through a nonstandard port that result in creating "holes" in corporate firewalls.
- Web services deviate from all these issues by relying on web standards or protocols instead of relying on proprietary protocols.
- Web services are based on the notion of a service and transfer the data in messages.
- A web service is a class that inherits from **System.Web.Services.WebService** and contains the method to be exposed with the **[WebMethod]** attribute over it.
- To access any remote components, you need a transport protocol, a message protocol, and the serialization mechanism for a client and server.
- In web services, the transport protocol is mainly HTTP, though it can be SMTP or TCP as well.
- As far as a message protocol is concerned, SOAP is the preferred and default message protocol. It also supports HTTP GET or HTTP POST. XML web services use XML serialization.

Sample XML Web Service Class

```
<%@ WebService Language="C#" Class="Order.Employee" %>
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
namespace Order
{
        public class Employee : WebService
        {
                [WebMethod()]
                public DataSet GetCustomer(int CustomerID)
                {
                // logic to retrieve customer
                }
                public DataSet DeleteCustomer(int CustomerID)
                {
                // logic to delete Orders of a customer
                }
        }
}
```

MERITS OF XML WEB SERVICE

- One of the major benefits is the Web services' ease of integration. You will easily integrate your services with other pieces of software. You can run on all kinds of machines, from the desktop to the mainframe, either within your enterprise or at external sites. This ease of integration will enable tighter business relationships and more efficient business processes
- With Web services readily available, and as the pool of XML Web services grows, you would be able to find software modules that can be integrated into your own application, by finding it and integrating it through XML Web services. Integrate with existing Web services instead of reinventing them. The bottom line is that you will be able to develop applications much faster than before.
- An integral part of the XML Web services programming model, is the ease of integration with external data sources. No longer does each application need to copy and maintain external data sources. You can request and get information in real time, and transform it to your particular format. This will allow you to deliver individualized software and services, while your maintenance burden is reduced.
- Consumers will enjoy ease of use when using XML Web services-based applications. XML Web services link applications, services, and devices together. Using Web services will be an integrated experience that excels in its simplicity. XML Web services give users the ability to act on information any time, any place, and from any smart device.

DEMERITS OF XML WEB SERVICE

- An XML Web service page contains a complete set of information that describes how the data will be formatted, how to wrap the data in a SOAP header, and how to prepare it to be sent. However, it doesn't tell you how to deliver it over the transports and to use a specific type of security. All of those transport-oriented notions are missing from today's ASMX services
- Another limitation of ASMX is the tight coupling with the HTTP runtime and the dependence on IIS to host it. This problem is solved by WCF, which can be hosted by any Windows process that is able to host the .NET Framework 3.0
- XML Web service is instantiated on a per-call basis, while WCF gives you flexibility. By providing various instancing options such as Singleton, private session, per call.
- XML Web service provides the way for interoperability but doesn't fulfill some of the basic requirements; for example, it does not provide or guarantee end to end security or reliable communication

Context:

- Understanding the need of xml web services
- Understanding the merits and Demerits of xml web services

Practice Session:

- Find out the advantages of XML Web Services over .Net Remoting, COM and DCOM

Common Errors:

- Non inclusion of [**WebMethod**] attribute
- Running the application without running web service at least once.

Lessons Learnt:

☑ Understanding the basics of xml web service
☑ Importance of [WebMethod] attribute
☑ Understanding the merits and demerits of XML Web service class

Topic: Overview of SOA

Estimated Time: 10 minutes.

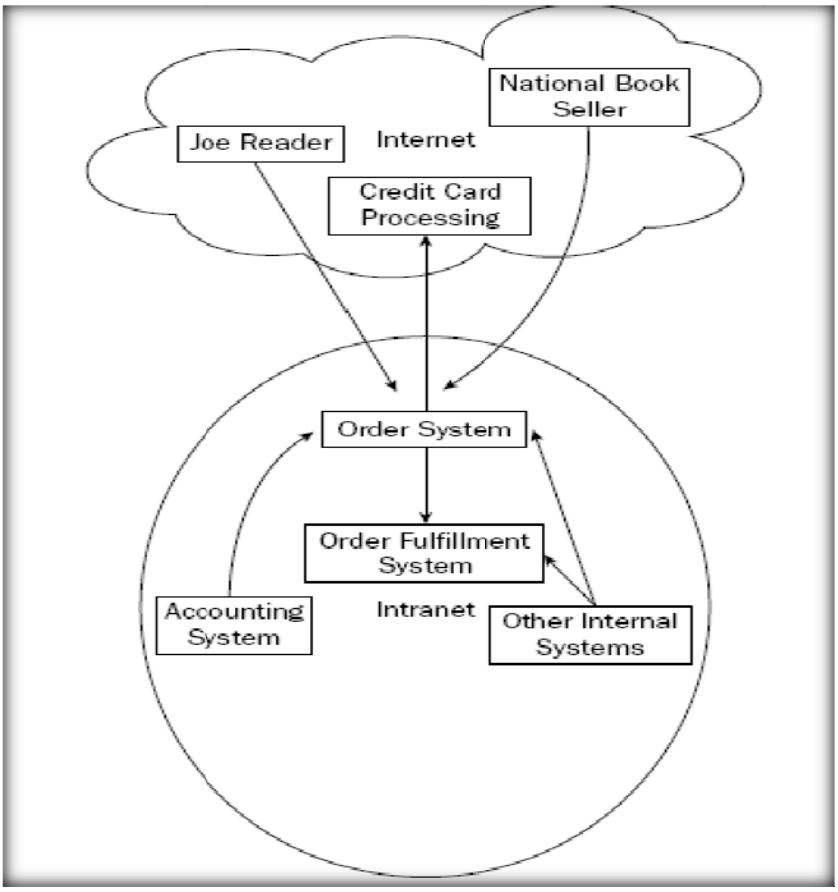
Objectives: This module will familiarize the participant with

- The need of Service Oriented Architecture

Presentation:

The Need for SOA

To understand Service-Oriented Architecture, take a look at the scenario shown in Figure



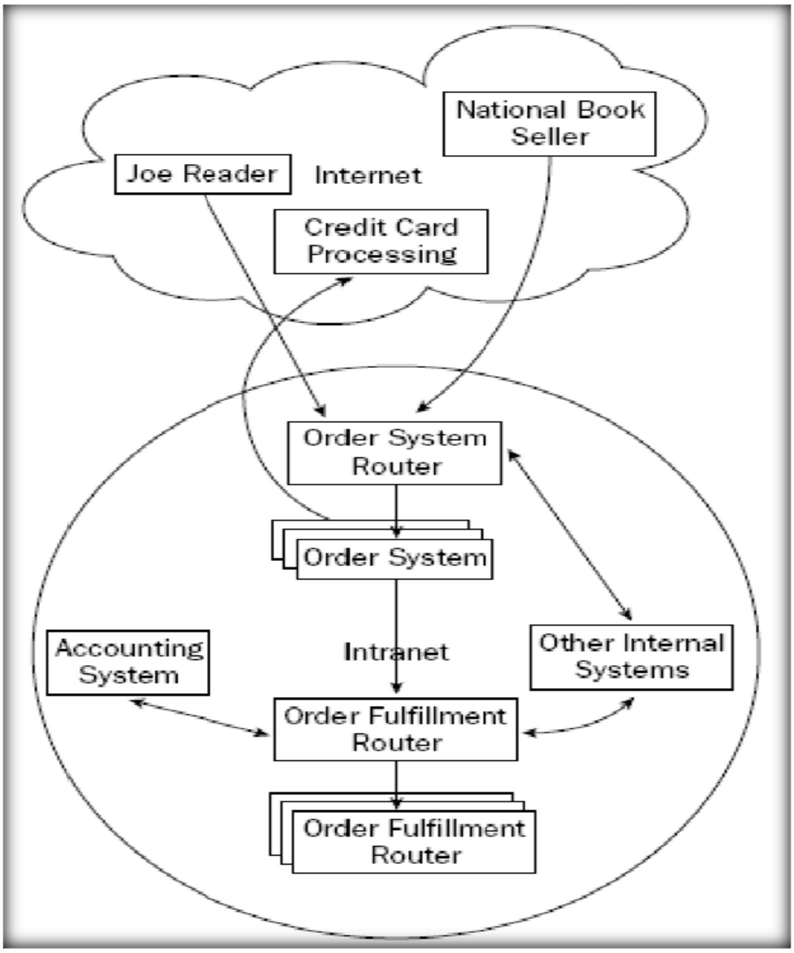
- In this example, a book publisher can receive book orders from both a single, individual reader as well as large quantity book orders from large national book resellers.
- Orders are received by the Order System application, which collects and processes the orders.

- Internally, the Order System application collects and processes the orders, such as validating credit cards and forwarding the order to the Order Fulfillment system. Both the Order System application and the Order Fulfillment application communicate with other internal applications and systems for various reasons
- Over time this publishing company becomes popular because it is hiring great authors and putting out high-quality books, and it becomes apparent that this simple solution is not keeping up with the demand and volume of orders.

- Service-Oriented Architecture says that the current system can, and should, be flexible enough to allow changes to the existing architecture without disrupting the current architecture and infrastructure currently in place. That is, each piece should be isolated enough that it can be replaced without disturbing the flow and process of the rest of the system. It is the concept of designing the technology processes within a business.

Understanding Service Orientation

Building this with service orientation in mind, several changes are made to the architecture, which fulfills the needs of the system without disrupting the process, as shown in Figure



- To handle the amount of orders coming in, a router was put in front of the Order Process service that then distributes the orders to one of many Order Process services.
- An Order Fulfillment router was also placed in front of the Order Fulfillment service, which accomplishes the same thing as the Order Process router; that is, it takes the incoming orders from the Order Process service and distributes the orders to one of many Order Fulfillment services.
- The other internal systems can still communicate and exchange information with these services without any changes.
- Externally, Joe Reader and the National Book Seller have no idea that changes were made at the publisher's end—it is all transparent to them. In fact, they might even see a better responding system when placing orders.
- The key here is that with SOA, major changes can take place behind the scenes completely transparent to the end user and without any interruption of the system.
- It is possible to have a web service that follows no SOA principles and a web service that exhibits wonderful SOA traits. For example, a good web service is almost self-describing, providing useful information. I want to hit a web service that gives me stock quotes, or lets me buy or sell stocks, or tells me how my portfolio is doing. In the case of the book publisher, I want a web service that tells me information about my order. These are web services that exhibit SOA traits.

Service-Oriented Architecture Principles

Streams of information have been flowing from Microsoft in the form of articles and white papers regarding its commitment to SOA, and in all of this information one of the big areas constantly stressed are the principles behind service orientation:

- Explicit boundaries
- Autonomous services
- Policy-based compatibility
- Shared schemas and contracts

Explicit Boundaries

- As you will learn in the next section, SOA is all about messaging—sending messages from point A to point B.
- These messages must be able to cross explicit and formal boundaries regardless of what is behind those boundaries. This allows developers to keep the flexibility of how services are implemented and deployed.
- Explicit boundaries mean that a service can be deployed anywhere and be easily and freely accessed by other services, regardless of the environment or development language of the other service.
- The thing to keep in mind is that there is a cost associated with crossing boundaries. These costs come in a number of forms, such as communication, performance, and processing overhead costs. Services should be called quickly and efficiently.

Autonomous Services

- Services are built and deployed independently of other services.
- Systems, especially distributed systems, must evolve over time and should be built to handle change easily. This SOA principle states that each service must be managed and versioned differently so as to not affect other services in the process.

- In the book publisher example, the Order Process service and Order Fulfillment service are completely independent of each other; each is versioned and managed completely independent of the other.

Policy-Based Compatibility

- When services call each other, it isn't like two friends meeting in the street, exchanging pleasantries, and then talking. Services need to know a little more about each other. Each service may or may not have certain requirements before it will start communicating and handing out information.
- Each service has its own compatibility level and knows how it will interact with other services. These two friends in fact aren't friends at all. They are complete and total strangers. When these two strangers meet in the street, an interrogation takes place, with each person providing the other with a policy.
- This policy is an information sheet containing explicit information about the person. Each stranger scours the policy of the other looking for similar interests. If the two services were to talk again, it would be as if they had never met before in their life. The whole interrogation process would start over. This is how services interact.
- Services look at each others' policy, looking for similarities so that they can start communicating. If two services can't satisfy each others' policy requirements, all bets are off. These policies exist in the form of machine-readable expressions. Policies also allow you to move a service from one environment to another without changing the behavior of the service.

Shared Schemas and Contracts

- Think "schemas = data" and "contracts = behavior." The contract contains information regarding the structure of the message.
- Services do not pass classes and types; they pass schemas and contracts. This allows for a loosely coupled system where the service does not care what type of environment the other service is executing on. The information being passed is 100 percent platform independent.

Context:

- Building Service Oriented Applications
- Understanding the need of SOA

Common Errors:

- Thinking SOA as of a language

Lessons Learnt:

☑ A loosely coupled architecture designed to meet the needs of the Organization

☑ Coarse granularity is one of the most important features of SOA

☑ SOA represents business functions as shared, reusable services

Topic: Introduction to WCF & its Architecture **Estimated Time: 15 minutes.**

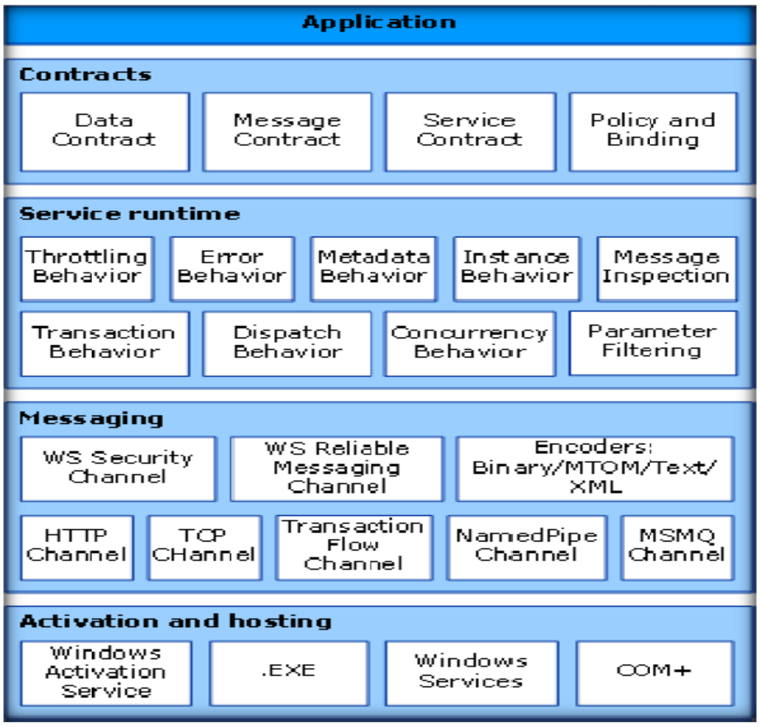
Objectives: **This module will familiarize the participant with**

- WCF basics.
- Role of WCF in distributed computing.
- WCF Architecture.
- Understanding WCF Architecture elements.

Presentation:

- Windows Communication Foundation (WCF) is a dedicated communication frame work provided by the Microsoft. WCF is a part of .NET 3.0.
- The runtime environment provided by the WCF enables us to expose our CLR types as services and to consume other existing services as CLR types.
- It is a fusion of current distributed system technologies designed and developed from day one with the goal of achieving SOA nirvana, or coming as close to it as possible.
- WCF is a programming model that enables developers to build service solutions that are reliable and secure, and even transacted.

Windows Communication Foundation Architecture



Contract and Different types of Contracts

- A contract I may sign could contain information such as the type of work I will perform and what information I might make available to the other party.
- It contains information that stipulates what a service does and the type of information it will make available.

Data Contract

- A data contract explicitly stipulates the data that will be exchanged by the WCF service.
- The WCF service and the client do not need to agree on the types, but they do need to agree on the data contract. This includes parameters and return types.

Message Contract

- A message contract provides additional control over that of a data contract, in that it controls the SOAP messages sent and received by the WCF service.
- A message contract lets you customize the type formatting of parameters in SOAP messages.

Service Contract

- A service contract is what informs the clients and the rest of the outside world what the endpoint has to offer and communicate.

Policy and Binding

- Policy and binding contracts specify important information such as security, protocol, and other information, and these policies are interrogated looking for the things that need to be satisfied before the two services start communicating.

Service Runtime

- The Service Runtime layer is the layer that specifies and manages the behaviors of the service that occur during service operation, or service runtime (thus “service runtime behaviors”).
- They have no control over endpoint or message behaviors. Likewise, endpoint and message behaviors have no control over service behaviors.

The following lists the various behaviors managed by the Service Runtime layer:

- **Throttling Behavior**: The Throttling behavior determines the number of processed messages.
- **Error Behavior**: The Error behavior specifies what action will be taken if an error occurs during service runtime.
- **Metadata Behavior**: The Metadata behavior controls whether or not metadata is exposed to the outside world.
- **Instance Behavior**: The Instance behavior drives how many instances of the service will be available to process messages.
- **Message Inspection**: Message Inspection gives the service the ability to inspect all or parts of a message.
- **Transaction Behavior**: The Transaction behavior enables transacted operations. That is, if a process fails during the service runtime it has the ability to rollback the transaction.
- **Dispatch Behavior**: When a message is processed by the WCF infrastructure, the Dispatch Behavior service determines how the message is to be handled and processed.

- **Concurrency Behavior**: The Concurrency behavior determines how each service, or instance of the service, handles threading. This behavior helps control how many threads can access a given instance of a service.
- **Parameter Filtering**: When a message is acted upon by the service, certain actions can be taken based on what is in the message headers. Parameter Filtering filters the message headers and executes preset actions based on the filter of the message headers.

Messaging

- The Messaging layer defines what formats and data exchange patterns can be used during WCF service communication.
- Client applications can be developed to access this layer and control messaging details and work directly with messages and channels.

The following lists the channels and components that the Messaging layer is composed of:

- **WS Security Channel**: The WS Security channel implements the WS-Security specification, which enables message security.
- **WS Reliable Messaging Channel**: Guaranteed message delivery is provided by the WS Reliable Messaging channel.
- **Encoders**: Encoders let you pick from a number of encodings for the message.
- **HTTP Channel**: The HTTP channel tells the service that message delivery will take place via the HTTP protocol.
- **TCP Channel:** The TCP channel tells the service that message delivery will take place via the TCP protocol.
- **Transaction Flow Channel**: The Transaction Flow channel governs transacted message patterns.
- **NamedPipe Channel**: The NamedPipe channel enables inter-process communication.
- **MSMQ Channel**: If your service needs to interoperate with MSMQ, this is the channel that enables that.

Activation and Hosting

- The Activation and Hosting layer provides different options in which a WCF service can be started as well as hosted.
- WCF Services can be hosted within the context of another application, or they can be self-hosted. This layer provides those options.

The following list details the hosting and activation options provided by this layer:

- Windows Activation Service: The Windows Activation Service enables WCF applications to be automatically started when running on a computer that is running the Windows Activation Service
- .EXE: WCF allows services to be run as executables (.EXE files).
- Windows Services: WCF allows services to be run as a Windows service.
- COM+: WCF allows services to be run as a COM+ application.

Context:

- To Understand the Windows Communication foundation

- To get familiarized with WCF Architecture

Lessons Learnt:

☑ WCF is a programming model that enables developers to build WCF service solutions that are reliable and secure, and even transacted.

☑ It is a mixture of current distributed system technologies designed and developed from day one with the goal of achieving SOA.

Topic: WCF Features

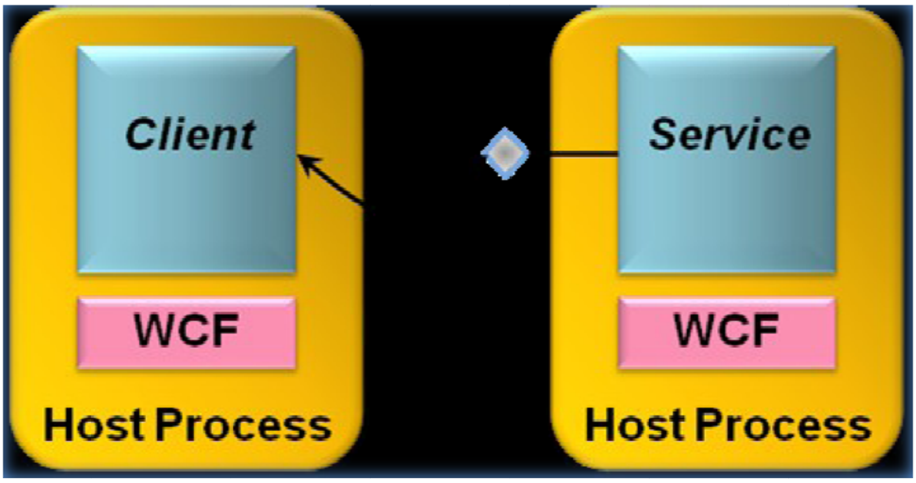
Estimated Time: 5 minutes.

Objectives: This module will familiarize the participant with

- Windows Communication Foundation Features.

Presentation:

- WCF is implemented primarily as a set of classes on top of the .NET Framework's Common Language Runtime (CLR).
- Because it extends their familiar environment, WCF allows .NET developers to build service-oriented applications in a familiar way.
- Both the client and the service can run in pretty much any Windows process—WCF doesn't define a required host. Wherever they run, clients and services can interact via SOAP, via a WCF-specific binary protocol, or in some other way.



Three things stand out, however, as WCF's most important aspects:

- Unification of existing .NET Framework communication technologies
- Interoperability with applications built on other technologies
- Explicit support for service-oriented development.

WCF Features:

- Because WCF can communicate using Web services, interoperability with other platforms that also support SOAP, such as Java EE application servers, is straightforward.
- Managing object lifetimes, defining distributed transactions, and other aspects of Enterprise Services are now provided by WCF. They are available to any WCF-based application,
- Because it supports a large set of the WS-* specifications, WCF helps provide reliability, security, and transactions when communicating with any platform that also supports these specifications.

- WCF's option for queued messaging, built on MSMQ, allows applications to use persistent queuing without needing to use another set of application programming interfaces.
- The version of WCF in the .NET Framework 3.5 has built-in support for creating restful clients and services.

Context:

- The Windows Communication Foundation (WCF) can be thought of as a messaging infrastructure.
- WCF Service operations can receive messages, process them and send them messages. Messages are described using operation contract.
- The Web Programming Model provides the basic framework elements required to build Web-style services with Windows Communication Foundation (WCF).
- Web-style services are designed to be accessed by the widest range of possible clients (including Web browsers with no additional client framework)

Exceptions:

- Trying to Access WCF service provided by framework 3.5, whereas the system supports .Net version 1.1.

Lessons Learnt:

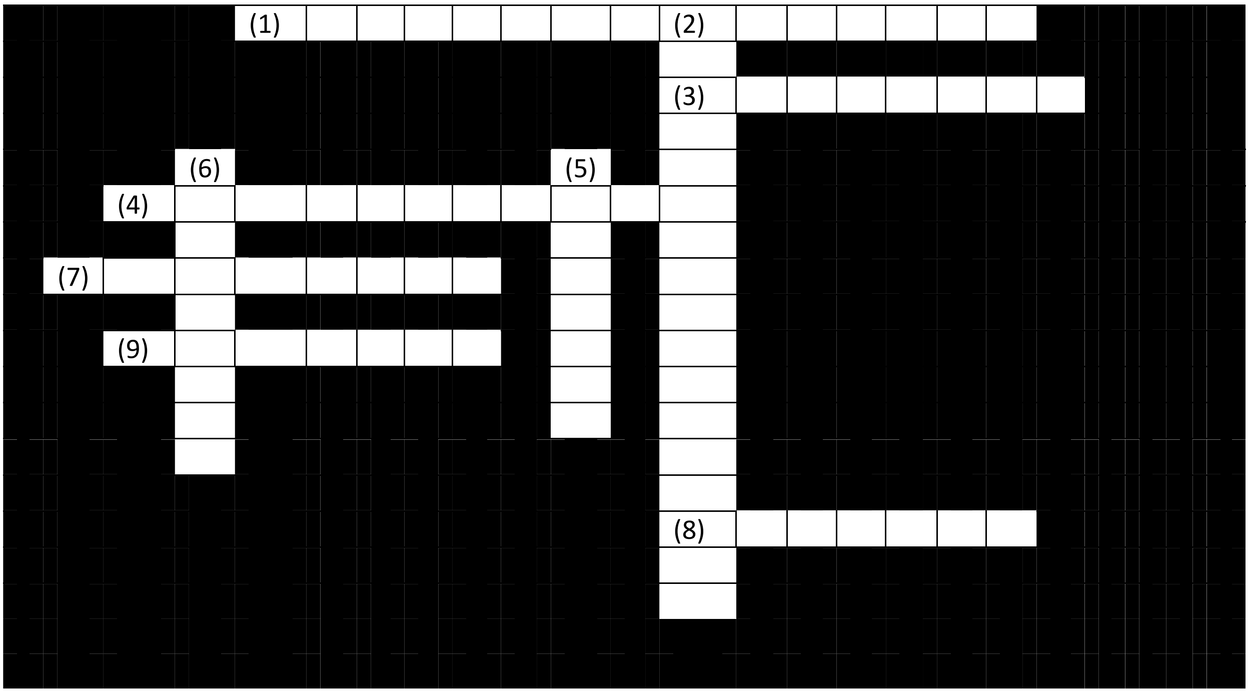
☑ Understanding the importance of using WCF
☑ Understanding the Features of WCF

Best Practices:

- Use WCF because it provides a single, unified and extendable programming object model that can be used to interact with a number of previously diverse distributed technologies.
- Use WCF service in application from security point of view.

Crossword-1

Estimated Time: 10 minutes.



ACROSS:

1. Developer uses WCF defined ______ attribute to determine which of its methods are exposed as client callable operation. **(15)**

3. It is single interface in which service can communicate with client. **(8)**

4. In order to host a WCF service in arbitrary process we create a new instance of one particular class. **(11)**

7. They are core abstraction for sending and receiving messages from an endpoint. **(8)**

8. Specifies where to find a WCF service. **(7)**

9. Specifies how client can communicate with endpoint including transport protocol, encoding and security requirements. **(7)**

DOWN:

2. The methods in WCF Service contract that can be invoked by client should be marked with _____ attribute. **(17)**

5. _____ is a collection of operations that specifies what endpoint can communicate to outside world. **(8)**

6. Are the types that modify or extend service or client functionality? **(9)**

2.0 The Basic Composition of WCF Service

Topics

- The ABCs of WCF
- Understanding of WCF Address
- Understanding of WCF Bindings
- Understanding of WCF Contracts
- Crossword

Topic: The ABCs of WCF

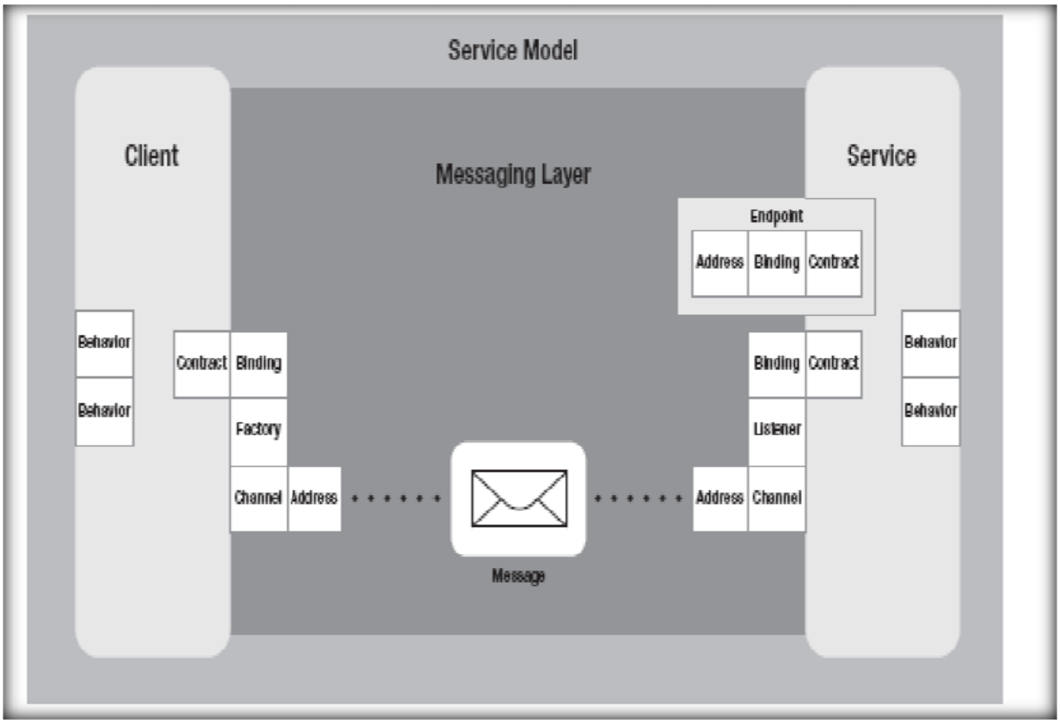
Estimated Time: 10 minutes.

Objectives: This module will familiarize the participant with

- Address, Binding and Contract components of WCF Services

Presentation:

- What are the ABCs of WCF? This is a common question that is asked of WCF lovers. In short, ABC stands for Address, Binding, and Contract:
  - The Address specifies where the messages can be sent (or where the service lives).
  - The Binding describes how to send the messages.
  - The Contract describes what the messages should contain.
- An endpoint acts as a “gateway” to the outside world. Usually you can refer to these three items as the endpoint of a service defined in WCF.
- The Web Services Description Language (WSDL) is meant to describe service endpoints in a standardized way.
- A WSDL file describes what a service can do, how a service can be accessed, and where the service can be found.
- Figure illustrates all the components to consume WCF services using the programming model and to use and extend the messaging layer.
- Together, the address, binding, and contract are commonly referred to as an endpoint.



- On the client side, you can see only one endpoint consisting of an address, binding, and contract.
- Think of the client as a piece inside your program that is able to communicate with a service, not your entire application.
- Commonly this is referred to as a proxy for that service. Of course, it is possible to have your client connect to multiple services by using multiple proxies with different endpoints (that is, connected to different services).
- The client side can have a specific behavior to do local configuration, such as concurrency, instancing, throttling, error handling, transaction control, security, and so on.
- The service side can have multiple endpoints. A service just sits around and waits for Messages to come in on its endpoints.

Context:

- To Understand the ABCs of the WCF Service
- Identify the concept of WSDL
- To have language interoperability

Practice Session:

- Identify the process involved using in WCF Service.

Lessons Learnt:

☑ The primary features of WCF Service ABCs

Topic: Understanding WCF Addresses

Estimated Time: 10 minutes.

Objectives: This module will familiarize the participant with

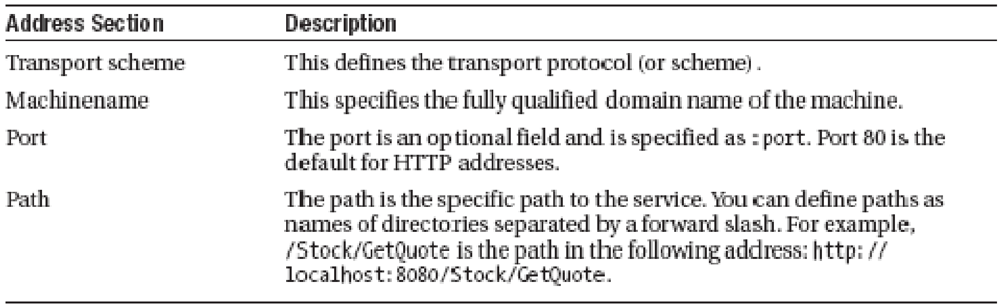
- Addresses in WCF Service configuration
- The attributes used while configuring Address section of WCF Service

Presentation:

What Are Addresses?

- Addressing a WCF service is essential to being able to use a service. You need to have the address of a WCF service to be able to send it a message.
- Addresses in WCF are URLs that define the protocol used, the machine where the service is running, and the path to the service. The port number is an optional field in the URL and depends on the protocol used.

Addressing Specifications



Address Section	Description
Transport scheme	This defines the transport protocol (or scheme).
Machinename	This specifies the fully qualified domain name of the machine.
Port	The port is an optional field and is specified as :port. Port 80 is the default for HTTP addresses.
Path	The path is the specific path to the service. You can define paths as names of directories separated by a forward slash. For example, /Stock/GetQuote is the path in the following address: http://localhost:8080/Stock/GetQuote.

- So, the format of a service address is as follows:
  scheme: //<machinename>[:port]/path1/path2
- WCF supports several protocols, and each has its own particular addressing format. WSAddressing is essential when WCF services use protocols other than HTTP.

Addressing HTTP:

- WCF Services can be hosted in different ways. HTTP services can be either self-hosted or hosted on Internet Information Services (IIS).
- When addressing an HTTP service in a self-hosted scenario, you use the following format:
  http://localhost:8080/QuickReturns/Exchange

When SSL is required, you can replace http with https. In a WCF configuration, you can set the HTTP address as follows:

```
<endpoint
        address="http://localhost:8080/QuickReturns/Exchange"
        bindingsSectionName="BasicHttpBinding"
        contract="IExchange" />
```

Addressing TCP

- The TCP transport uses the net.tcp: scheme but otherwise follows the same rules as described
  net.tcp://localhost:8080/QuickReturns/Exchange

In a WCF configuration, you can set the net.tcp address as follows:

```
<endpoint
          address="net.tcp://localhost:8080/QuickReturns/Exchange"
          bindingsSectionName="NetTcpBinding"
          contract="IExchange"
/>
```

Addressing MSMQ

- You can use the Microsoft Message Queue (MSMQ) transport in an asynchronous one-way (fire and- forget) or duplex type of messaging pattern and use the MSMQ features of Windows.
- MSMQ has public and private queues. Public queues are usually available through Active Directory and can be accessed remotely whereas private queues are local queues that are available only on the local machine.
- MSMQ addresses use the net.msmq scheme and specify a machine name, queue type, and queue name. Port numbers don't have any meaning in the MSMQ address, so a sample MSMQ address is as follows:
  net.msmq://localhost/private$/QuickReturnSettleTrade

In a WCF configuration, you can set the net.msmq address as follows:

```
<endpoint
       address=" net.msmq://localhost/private$/QuickReturnsSettleTrade"
       bindingsSectionName="NetMsmqBinding"
       contract="IExchange"
/>
```

Addressing Named Pipes

- Named Pipes is a common way to provide a means to implement inter- or in-process communication.
- The Named Pipes transport in WCF supports only local communication and uses the net.pipes scheme. Port numbers don't have any meaning with the Named Pipes transport.
- This results in the following address format:
  net.pipe://localhost/QuickReturns/Exchange
- In a WCF configuration, you can set the net.pipe address as follows:

```
<endpoint
        address="net.pipe://localhost/QuickReturns/Exchange"
        bindingsSectionName="NetNamedPipeBinding"
        contract="IExchange" />
```

Base Addresses

- WCF supports base addresses, which enables you to host multiple endpoints under the same base address and which shortcuts the duplication of the scheme, host, port, and root path in your configuration.

To define two endpoints in a WCF configuration, you would add the following section to express base address of QuickReturns Ltd:

```
<host>
        <baseAddresses>
        <add baseAddress="http://localhost:8080/QuickReturns"/>
        <add baseAddress="net.tcp://localhost/QuickReturns"/>
        </baseAddresses>
</host>
```

This allows you to define the following endpoints:

```
<endpoint
        name="BasicHttpBinding"
        address="Exchange"
        bindingsSectionName="BasicHttpBinding"
        contract="IExchange"
 />

<endpoint
        name="NetNamedPipeBinding"
        address="Exchange"
        bindingsSectionName="NetNamedPipeBinding"
        contract="IExchange"
/>
```

Context:

- Each WCF service will be associated with unique address.
- Address specifies where on the network to find WCF Service

Practice Session:

- Explore more about the SOAP Protocol

Check list:

- Importance of associating WCF with unique address.
- Different components of WCF Addresses

Lessons Learnt:

☑ Different components of WCF Address

☑ Different ways of using WCF Address

Topic: Understanding of WCF Bindings

Estimated Time: 10 minutes.

Objectives: This module will familiarize the participant with

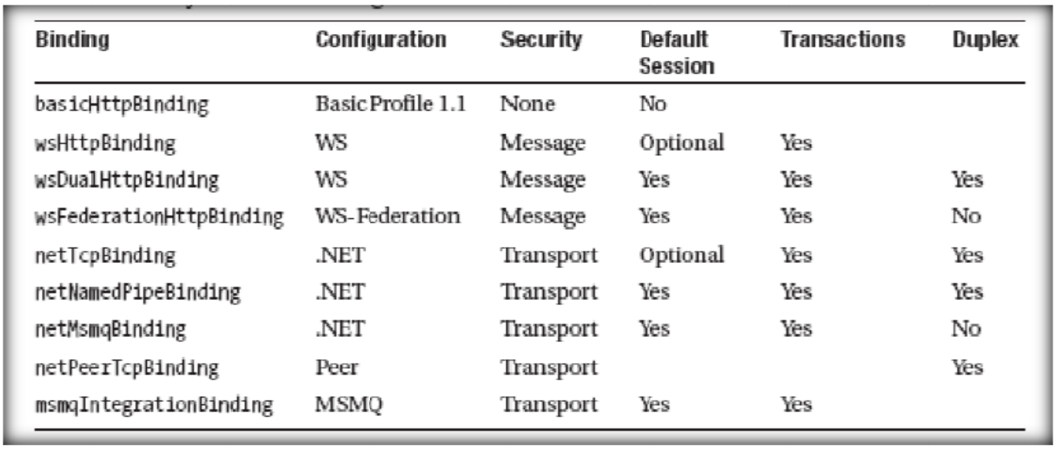
- The importance of Bindings in WCF Service configuration
- Components of WCF Bindings

Presentation:

What Are Bindings?

- A binding defines how you can communicate with the WCF service and as such has the biggest impact in the programming model of WCF. It is the primary extension point of the ABCs of WCF. The binding controls the following:
  - The transport (HTTP, MSMQ, Named Pipes, TCP)
  - The channels (one-way, duplex, request-reply)
  - The encoding (XML, binary, MTOM…)
  - The supported WS-* protocols (WS-Security, WS-Federation, WS-Reliability, WS-Transactions)

Predefined WCF Bindings



Binding	Configuration	Security	Default Session	Transactions	Duplex
basicHttpBinding	Basic Profile 1.1	None	No		
wsHttpBinding	WS	Message	Optional	Yes	
wsDualHttpBinding	WS	Message	Yes	Yes	Yes
wsFederationHttpBinding	WS-Federation	Message	Yes	Yes	No
netTcpBinding	.NET	Transport	Optional	Yes	Yes
netNamedPipeBinding	.NET	Transport	Yes	Yes	Yes
netMsmqBinding	.NET	Transport	Yes	Yes	No
netPeerTcpBinding	Peer	Transport			Yes
msmqIntegrationBinding	MSMQ	Transport	Yes	Yes	

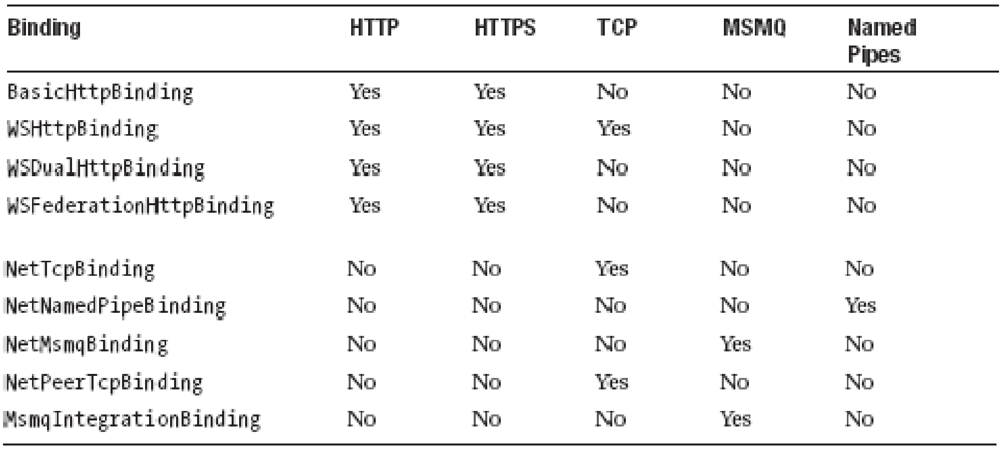
Remember, you can have multiple endpoints defined for a service so that your service supports any combination of these bindings. WCF supports several transports on the Microsoft platform:

- HTTP(S)
- TCP
- Named Pipes
- MSMQ

- Obviously, only HTTP(S) is truly an interoperable transport. When integration is required with different platforms, you can recognize interoperable bindings with the WS prefix.

- BasicHttpBinding is also an interoperable binding that maps very well on the pre-WCF service stacks such as ASMX.
- The bindings prefixed with Net are really Windows-Centric; where it is expected that interoperability is not a requirement.
- Table lists the predefined WCF bindings and the transport(s) they support.

Predefined WCF Bindings Mapped on the Transports



Binding	HTTP	HTTPS	TCP	MSMQ	Named Pipes
BasicHttpBinding	Yes	Yes	No	No	No
WSHttpBinding	Yes	Yes	Yes	No	No
WSDualHttpBinding	Yes	Yes	No	No	No
WSFederationHttpBinding	Yes	Yes	No	No	No
NetTcpBinding	No	No	Yes	No	No
NetNamedPipeBinding	No	No	No	No	Yes
NetMsmqBinding	No	No	No	Yes	No
NetPeerTcpBinding	No	No	Yes	No	No
MsmqIntegrationBinding	No	No	No	Yes	No

Practice Session:

- Explore more about MSMQ and NamedPipe protocols

Check list:

- Different components of Bindings

Common Errors:

- Trying to use the predefined bindings with the protocol which is not supported

Lessons Learnt:

☑ WCF Service Bindings

☑ Predefined WCF Bindings

☑ Composition of WCF Bindings

Topic: Understanding of WCF Contracts

Estimated Time: 15 minutes.

Objectives: This module will familiarize the participant with

- The importance of Contract in WCF Service configuration
- Components of WCF Contracts

Presentation:

- Contracts in Windows Communication Foundation provide the interoperability they need to communicate with the client.
- It is through contracts that clients and WCF services agree as to the types of operations and structures they will use during the period that they are communicating back and forth.

Three basic contracts are used to define a Windows Communication Foundation service:

- **Service Contract:** Defines the methods of a service, that is, what operations are available on the endpoint to the client.
- **Data Contract:** Defines the data types used by the available service methods.
- **Message Contract:** Provides the ability to control the message headers during the creation of a message.

Service Contract and [ServiceContract] Attribute:

- Service contract defines the operations, or methods, that are available on the service endpoint and is exposed to the outside world.
- A service contract exposes specific information to the client, which enables the client to understand what the service has to offer. This information includes the following:

The data types in the message

- The locations of the operations
- The protocol information and serialization format to ensure the proper and successful communication
- Operation grouping
- Message exchange pattern (MEPs)

A service contract is defined by simply applying the [ServiceContract] annotation to an interface or class.

The following example shows how to define an interface as a service contract:

```
[ServiceContract]
public interface ICalculator
{
        //declare interface members here
}
```

WCF Service operations are specified by applying the [OperationContract] annotation on the methods of an interface, as illustrated in this example:

```
 [OperationContract]
int FindSum(int x,int y)

[OperationContract]
int  FindDifference(int x,int y)
```

Put these two together to make a complete service contract, as shown below:

```
[ServiceContract]
public interface ICalculator
{
            [OperationContract]
            int FindSum(int x,int y);
            [OperationContract]
            int  FindDifference(int x,int y);
}
```

Once the interface for the WCF service is defined, the defined interface can then be implemented. The following example implements the IBookOrder interface defined in the preceding code:

```
public class Calculator : ICalculator
{
   public int sum(int x,int y)
   {
                return (x + y);
   }
   public int diff(int x,int y)
   {
                return (x - y);
   }
}
```

[OperationContract] Attribute:

- The [OperationContract] attribute includes the method as part of the service contract and identifies the method as a WCF service operation.
- A method marked as an operation contract is exposed to the public. Methods not marked with this attribute are not exposed externally.
- [OperationContract] attributes are also mapped to an equivalent WSDL operation definition.

Data Contracts:

- Simply, a data contract describes the data that is to be exchanged. Before these two strangers can start talking, they each need to agree on the “data” that they will be passing back and forth. This is a data contract. It is a formal agreement between the two parties that contains information regarding the data that they will be exchanging.

[DataContract] Attribute:

- Data contracts are defined declaratively just like service contracts and operation contracts. To define a data contract, you simply decorate a class or enumeration with the [DataContract] attribute

[DataMember] Attribute:

- The [DataMember] attribute is applied to all members of the data contract type to identify it as a data member. Specifying this attribute identifies that member as member whose data will be serialized by the DataContractSerializer

Message Contracts:

- Message contracts provide the ultimate control over the formatting of the SOAP messages. In most cases you will not need to employ this level because data contracts provide most of the control you will need.
- A message contract provides the level of interoperability you would need when you need to communicate with clients or other systems that may use a particular WSDL or schema, or cases when there is some question or concern regarding the SOAP message structure, such as controlling security issues.
- This section discusses the [MessageContract], [MessageHeader], and [MessageBodyMember] attributes along with their associated parameters.

[MessageContract] Attribute:

- Message contracts are defined by applying the [MessageContract] attribute.
- You then specify specifics for each message using the [MessageBodyMember] and [MessageHeader] attributes. The following example illustrates these three attributes by defining a simple message contract:

```
[MessageContract]
  public class BuyStock
  {
            [MessageHeader]
            public string Title;
            [MessageBodyMember]
            public decimal Cost;
  }
```

- The [MessageContract] attribute defines a strongly typed class that is associated with a SOAP message. This in turn creates a typed message, which then enables the passing of messages between WCF services (service operations). It also provides the ability to use custom messages as parameters within these same WCF services (and service operations).
- The [MessageHeader] attribute maps a SOAP message header to fields and properties of a type marked with the [MessageHeader] attribute.
- The fields or properties can be a simple or composite type that can be serialized

Practice Session:

- Find out the various properties of WCF Contracts.

Check list:

- Different types of WCF Contracts
- Service Contract
- Data Contract
- Message Contract

Common Errors:

- Trying to use the predefined bindings with the protocol which is not supported

Lessons Learnt:

☑ Contracts in Windows Communication Foundation provide the interoperability they need to communicate with the client.

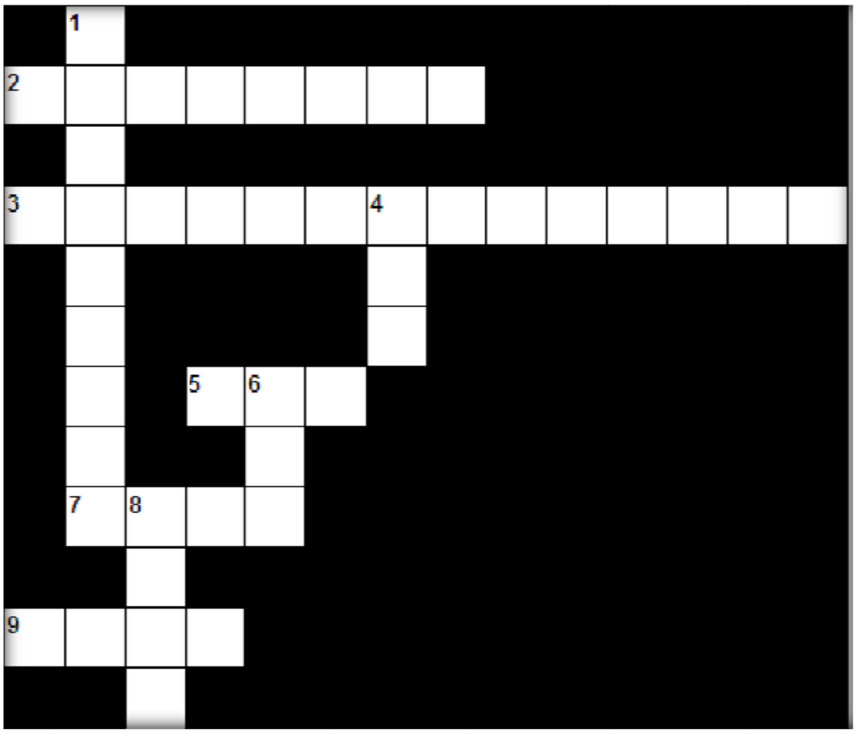
☑ Service contract defines the operations, or methods, that are available on the service endpoint and is exposed to the outside world

☑ Data contract describes the data that is to be exchanged.

☑ Message contracts provide the ultimate control over the formatting of the SOAP messages.

Crossword-2

Estimated Time: 10 minutes.



ACROSS:

2. It solves problems of versioning, reference counting which the major drawbacks in COM & DCOM were (8)
3. It is component on web that is accessible through open standards such as SOAP & HTTP (14)
5. It is unified programming model which helps in building service oriented applications (3)
7. It is a proprietary wire protocol standard from Microsoft to extend COM so it can work in distribute environment (4)
9. It is set of objects which allow you to perform queue messaging in windows? (4)

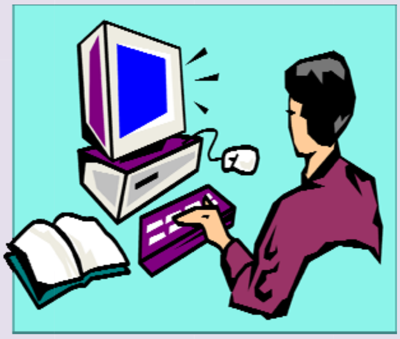
DOWN:

1. It is attribute which exposes the methods of xml web services (9)
4. Code granularity is one of the important features of it (3)
6. To enable applications to interact with each other and promote reusability (3)
8. It makes your applications more suitable by providing thread pooling, object pooling (4)

3.0 Building, Hosting and Consuming a WCF Service

Topics

- Building WCF Service
- Building Host for WCF Service
- Building Client Application which consume WCF Service
- Crossword



Topic: Building WCF Service

Estimated Time: 30 minutes.

Objectives: This module will familiarize the participant with

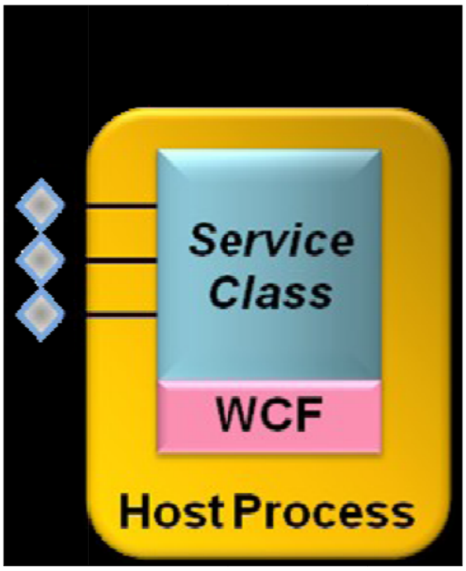
- Creating WCF Service using WCF Service Library template in Visual Studio.
- Testing WCF Service using WCF Test Client tool which integrated with Visual Studio.

Presentation:

- WCF provides a unified framework for rapidly building service-oriented applications that makes it easy to build and consume secure, reliable and transacted web services.
- It unifies the capabilities in ASMX, WSE, Remoting, COM+ and MSMQ; that for developers need to learn only one programming model.
- WCF simplifies development of connected applications through a new service-oriented programming model. WCF supports many styles of distributed application development by providing a layered architecture
- Windows Communication Foundation (WCF) is designed to offer a manageable approach to distributed computing, broad interoperability, and direct support for service orientation.

Creating a WCF Service

- As the figure below shows, every WCF service has three primary components:
- A service class, implemented in C# or Visual Basic or another CLR-based language that implements one or more methods.
- A host process in which the service runs.
- One or more endpoints that allow clients to access the WCF service. All communication with a WCF service happens via the service's endpoints.



- Understanding WCF requires grasping all these concepts. This section describes each one, beginning with service classes

Implementing a WCF Service Class

- A WCF service class is a class like any other, but it has a few additions. These additions allow the class's creator to define one or more contracts that this class implements.
- Each service class implements at least one service contract, which defines the operations this service exposes.
- The class might also provide an explicit data contract, which defines the data those operations convey.

Defining Service Contracts

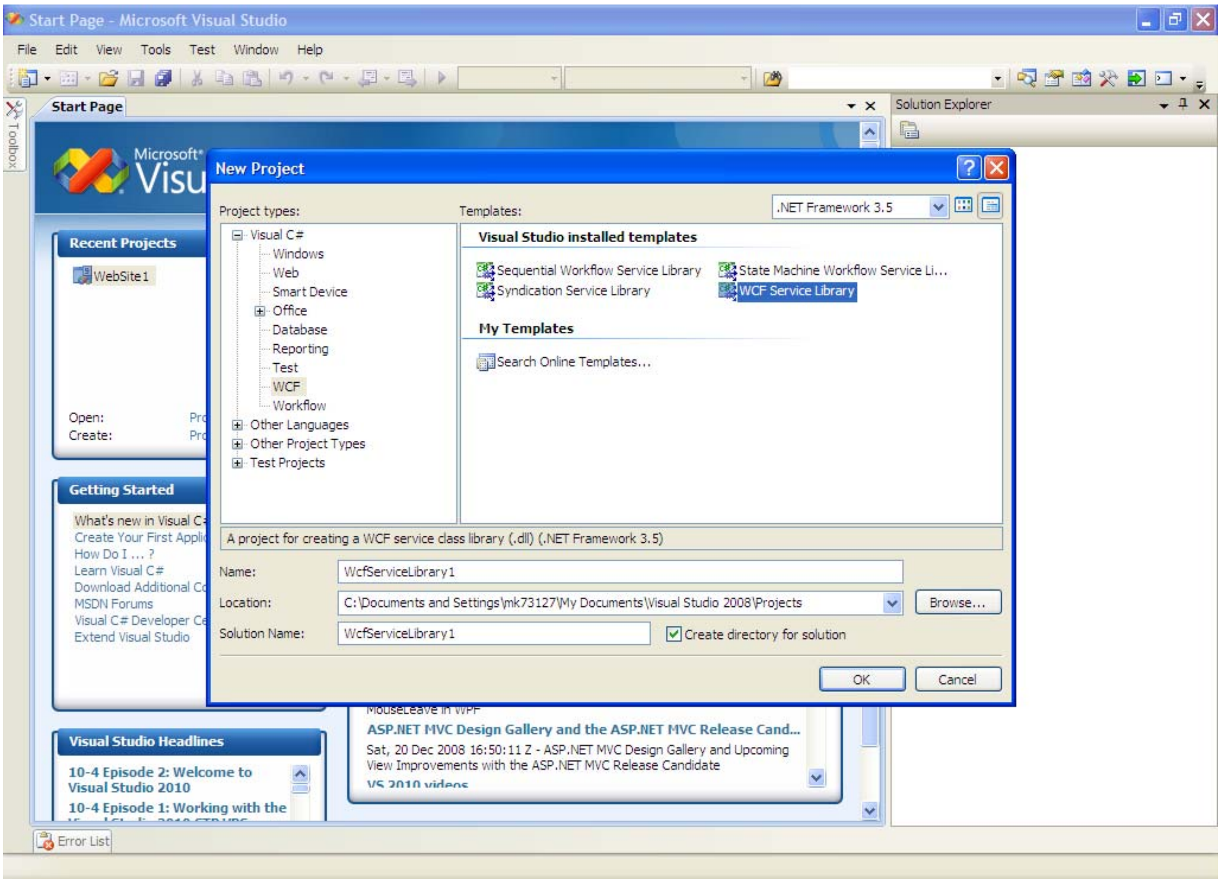
- Every WCF service class implements methods for its clients to use.
- The creator of the class determines which of its methods are exposed as client-callable operations by specifying that they are part of some service contract.
- To do this, a developer uses the WCF-defined attribute ServiceContract. In fact, a service class is just a class that either is itself marked with the ServiceContract attribute or implements an interface marked with this attribute.

Scenario:

Mr. Cristino is working as component developer; he got a requirement for creating a component/service which provides all the functionality for doing standard arithmetic calculations. Mr. Cristono decided to create a WCF service having this functionality, which in turn can be reused in any type of application and on any platform.

Demonstration/Code Snippet:

1. On the File menu, click New Project.
2. In the New Project dialog box, expand the Visual C# node and select WCF, and then select WCF Service Library. Click OK to open the project.



Start Page - Microsoft Visual Studio
File Edit View Tools Test Window Help
Start Page
Solution Explorer
Recent Projects
WebSite1
Open:
Create:
Getting Started
What's new in Visual C#
Create Your First Appli
How Do I ... ?
Learn Visual C#
Download Additional C
MSDN Forums
Visual C# Developer C
Extend Visual Studio
Visual Studio Headlines
10-4 Episode 2: Welcome to Visual Studio 2010
10-4 Episode 1: Working with the
ASP.NET MVC Design Gallery and the ASP.NET MVC Release Cand...
Sat, 20 Dec 2008 16:50:11 Z - ASP.NET MVC Design Gallery and Upcoming View Improvements with the ASP.NET MVC Release Candidate
Error List
New Project
Project types:
Visual C#
Windows
Web
Smart Device
Office
Database
Reporting
Test
WCF
Workflow
Other Languages
Other Project Types
Test Projects
Templates:
.NET Framework 3.5
Visual Studio installed templates
Sequential Workflow Service Library
State Machine Workflow Service Li...
Syndication Service Library
WCF Service Library
My Templates
Search Online Templates...
Browse...
OK
Cancel

A project for creating a WCF service class library (.dll) (.NET Framework 3.5)

Name: WcfServiceLibrary1

Location: C:\Documents and Settings\mk73127\My Documents\Visual Studio 2008\Projects

Solution Name: WcfServiceLibrary1

☑ Create directory for solution

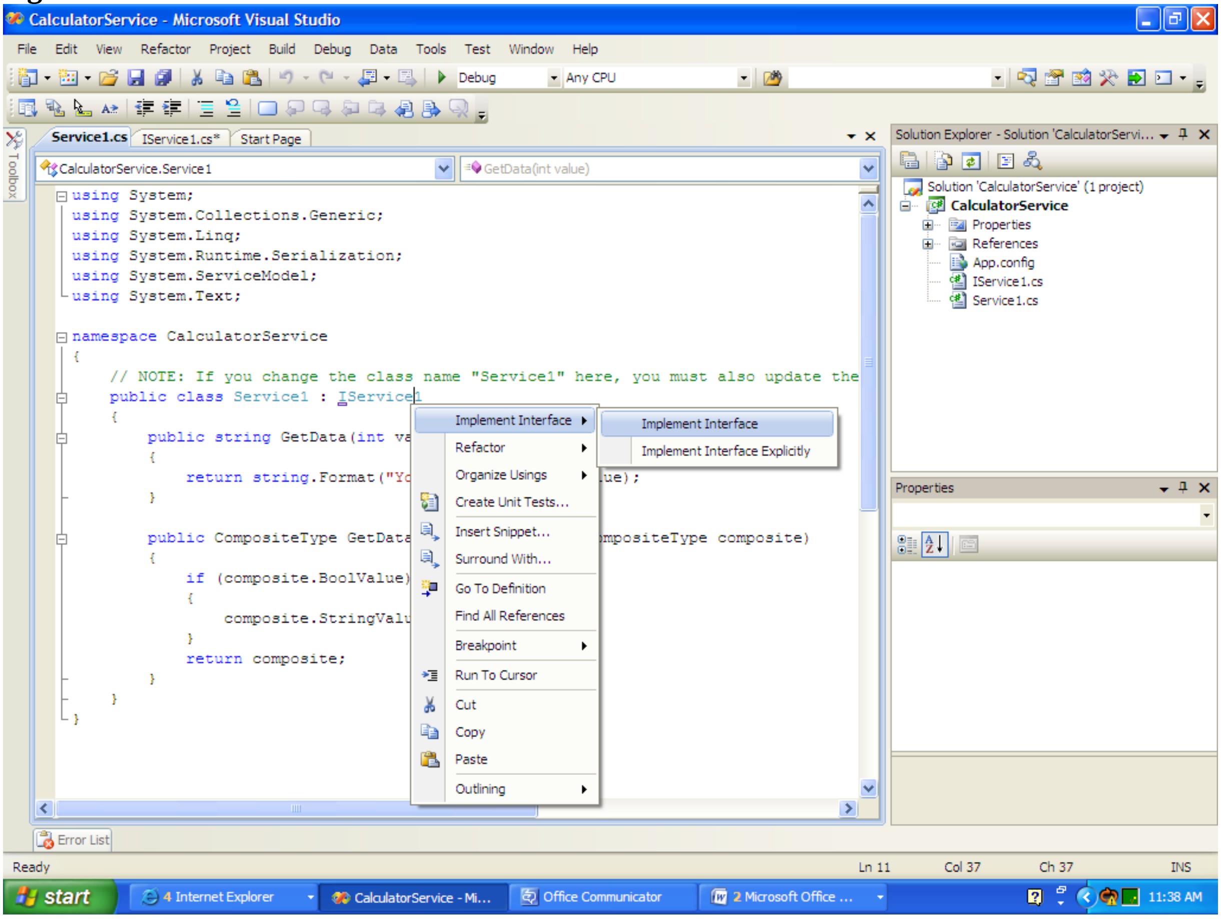
3. Change the name of application to CalculatorService
4. In Solution Explorer double click IService.cs and write the following code

Code for IService.cs Interface File:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace CalculatorService
{
    // NOTE: If you change the interface name "IService1" here, you must also update the
reference to "IService1" in App.config.
    [ServiceContract]
    public interface IService1
    {
        [OperationContract]
         int sum(int x, int y);
        [OperationContract]
         int diff(int x, int y);

        [OperationContract]
         int mul(int x, int y);

        [OperationContract]
         int div(int x, int y);

        // TODO: Add your service operations here
    }

    // Use a data contract as illustrated in the sample below to add composite types to service
operations
}
```

5. Double click on Service.cs and right click on the name of the interface and then click on the implement interface as shown in the figure below

Figure of the Service.cs Class File:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace CalculatorService
{
    // NOTE: If you change the class name "Service1" here, you must also update the
    public class Service1 : IService1
    {
        public string GetData(int va
        {
            return string.Format("Yo                     ue);
        }

        public CompositeType GetData                     mpositeType composite)
        {
            if (composite.BoolValue)
            {
                composite.StringValu
            }
            return composite;
        }
    }
}
```

6. Then write the functionality for all the methods which are initialized in the interface class IService.cs with the code shown below in Service.cs Class

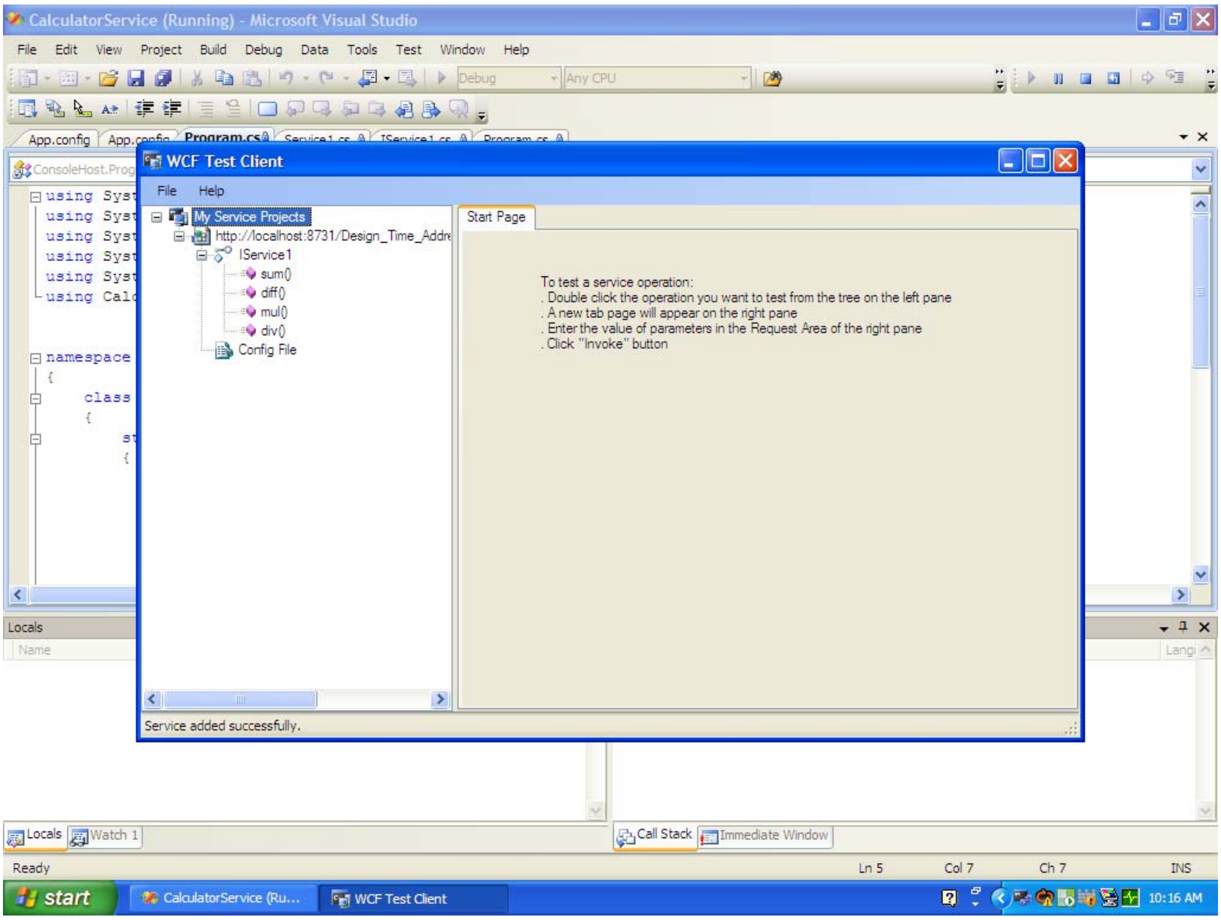
Code for the Service.cs Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace CalculatorService
{
    // NOTE: If you change the class name "Service1" here, you must also update the reference to "Service1" in App.config.
    public class Service1 : IService1
    {
        #region IService1 Members
        public int sum(int x, int y)
        {
            return (x + y);
        }
        public int diff(int x, int y)
        {
            return (x - y);
        }
        public int mul(int x, int y)
        {
            return (x * y);
        }
        public int div(int x, int y)
        {
            if (y == 0)
            {
                return 0;
            }
            else
            {
                return (x / y);
            }
        }
        #endregion
    }
}
```

Testing the WCF Service

WCF Test Client tool is one more .Net utility shipped along with .Net 3.0/3.5 framework. This tool can be used to test the WCF services either through command prompt (using WCFTestClient.exe command) or through Visual Studio 2008.

1. Press F5 to run the service. A WCF Test Client form will be displayed and it will load the WCF service.

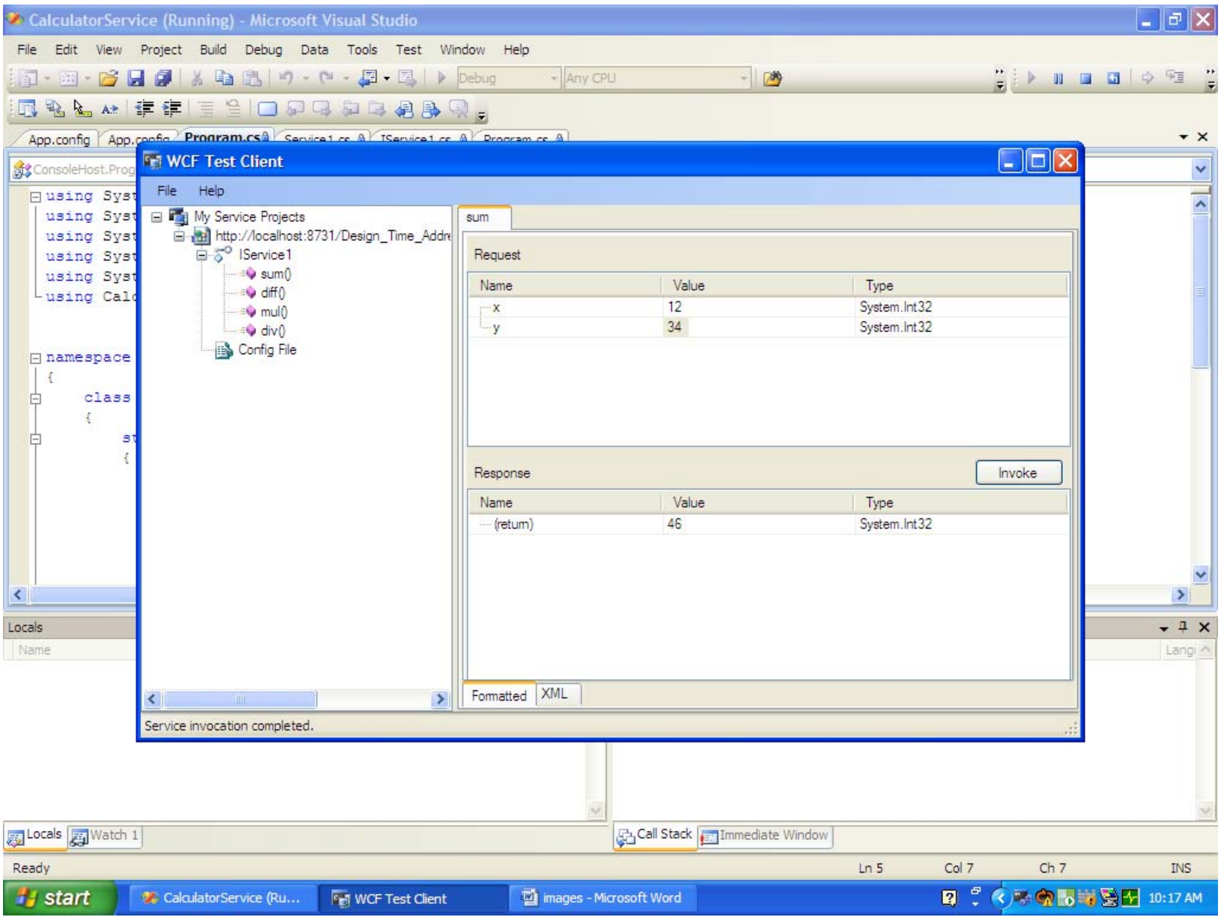


```
My Service Projects
  http://localhost:8731/Design_Time_Addre
    IService1
      sum()
      diff()
      mul()
      div()
    Config File
```

To test a service operation:
. Double click the operation you want to test from the tree on the left pane
. A new tab page will appear on the right pane
. Enter the value of parameters in the Request Area of the right pane
. Click "Invoke" button

Service added successfully.

2. In the WCF Test Client form, double-click the sum( ),diff( ) ,mul( ) and div( ) one at a time under IService1. Tab for providing values to method parameters will be displayed.
3. In the Request box, select the Value field and enter the values of two numbers x and y.
4. Click the Invoke button. If a Security Warning dialog box is displayed, click OK. The result will be displayed in the Response box.



My Service Projects
http://localhost:8731/Design_Time_Addre
IService1
sum()
diff()
mul()
div()
Config File

Request

Name	Value	Type
x	12	System.Int32
y	34	System.Int32

Response

Name	Value	Type
(return)	46	System.Int32

Service invocation completed.

5. In the File menu, click Exit to close the test form.

Notes:

- The ServiceContract attribute and all the other attributes that this example uses are defined in System.ServiceModel namespace, so the code begins with using statement that references this namespace.
- Each method in a WCF service class that can be invoked by a client must be marked with another attribute named OperationContract. All the methods in a WCF service class that are preceded by the OperationContract attribute are automatically exposed by WCF as services.
- In this example, sum, diff, mul, div are all marked with this attribute, so all four are exposed to clients of this WCF service. Any methods in a WCF service class that aren’t marked with OperationContract, aren’t included in the service contract, and so can’t be called by clients of this WCF service.

Practice Session:

- After exploring the usage of DataContract and MessageContract, enhance the above created WCF service by implementing both the contracts in relevant place.

Common Errors:

- Non inclusion on System.ServiceModel namespace in source code files.

Lessons Learnt:

☑ How to build a WCF Service
☑ Testing WCF Service

Topic: Building Host for WCF Service

Estimated Time: 30 minutes.

Objectives: **This module will familiarize the participant with**

- Hosting WCF Service
- Different ways of hosting WCF Service

Presentation:

Selecting a Host

- A WCF service class is typically compiled into a library.
- By definition, all libraries need a host Windows process to run in. WCF provides two main options for hosting libraries that implement services.
- One is to use a host process provided by either Internet Information Services (IIS) or a related technology in Windows Vista called the Windows Activation Service (WAS).
- The other allows a service to be hosted in an arbitrary process. The following section describes both options, beginning with the IIS/WAS approach.

Hosting a WCF Service Using IIS or WAS

- The simplest way to host a WCF service is to rely on IIS or WAS.
- Both rely on the notion of a virtual directory, which is just a shorter alias for an actual directory path in the Windows file system.
- To see how hosting with IIS and WAS works, suppose either of the CalculatorService classes shown earlier was compiled into the library reserve.dll, then placed in the virtual directory calculator on a system running Windows Server 2003.
- To indicate that the WCF service implemented in reserve.dll should be hosted by IIS or WAS, a developer creates a file in the calculator virtual directory with the extension .svc (which stands, of course, for “service”). For our simple example, this file might be called calculator.svc, and its entire contents could be:

```
<%@Service language=c# class="CalculatorService" %>
```

- Once this has been done and an endpoint has been defined as shown in the next section, a request from a client to one of the CalculatorService service’s methods will automatically create an instance of this class to execute the specified operation.That instance will run in standard process that IIS or WAS provides.
- Because it provides a standalone activation service, WAS allows hosting WCF services without running a full-blown web server. And while hosting WCF services in IIS looks just like hosting them in WAS, as shown above, there are a couple of significant differences:
- IIS-hosted WCF services can only be accessed using SOAP over HTTP. No other transport protocols are supported.
- Although WAS doesn’t require a Web server to be installed on the system, WCF services hosted in IIS obviously do.
- Whatever choice is made, both WAS and IIS provide WCF services with a range of support, such as the ability to configure automatic process recycling.

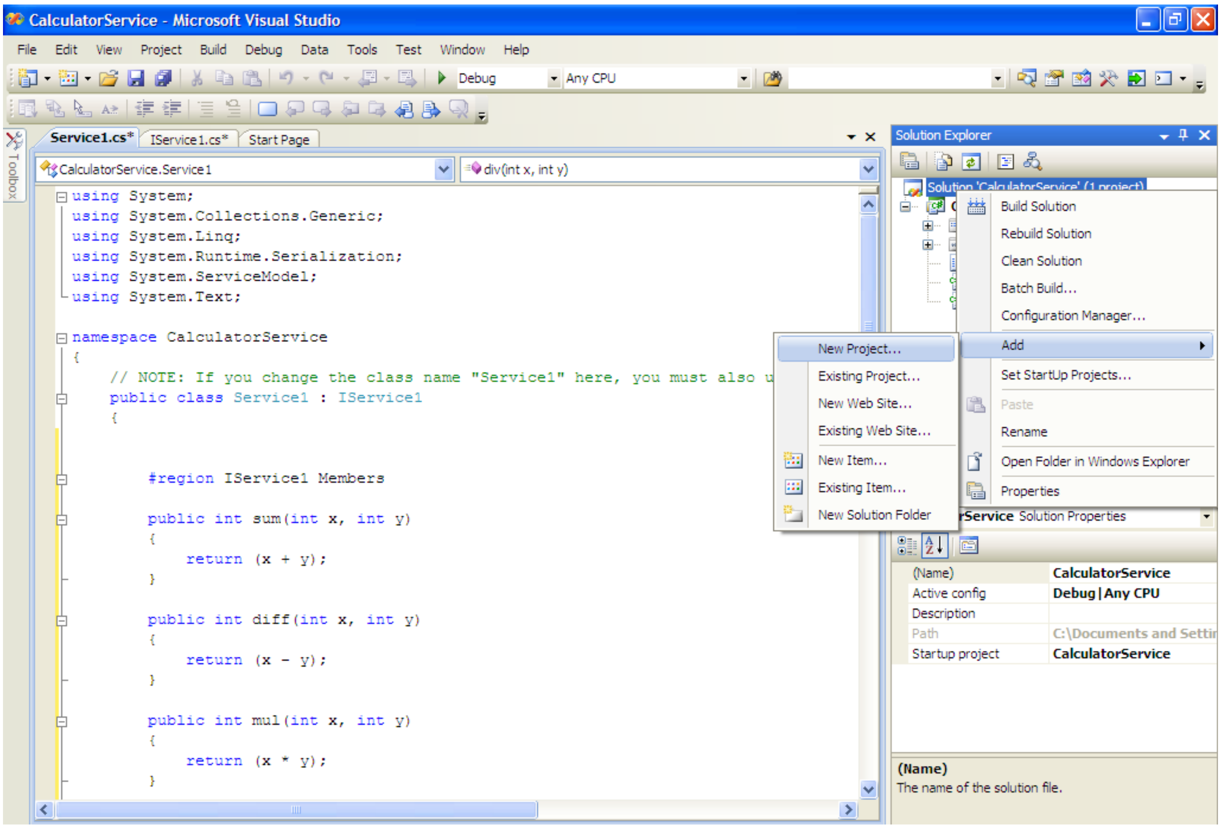
Demonstration/Code Snippet:

Hosting a WCF Service in an Arbitrary Process

- Relying on IIS or WAS to provide a process for hosting a WCF service is certainly the simplest choice.
- Yet applications often need to expose WCF services from their own process rather than relying on one provided by Windows.

The following example shows how to create a simple console application that hosts either of the CalculatorService classes defined earlier:

1. **Right click on the solution explorer of the existing CalculatorService project, add new project and select console application and name it as ConsoleHost.**

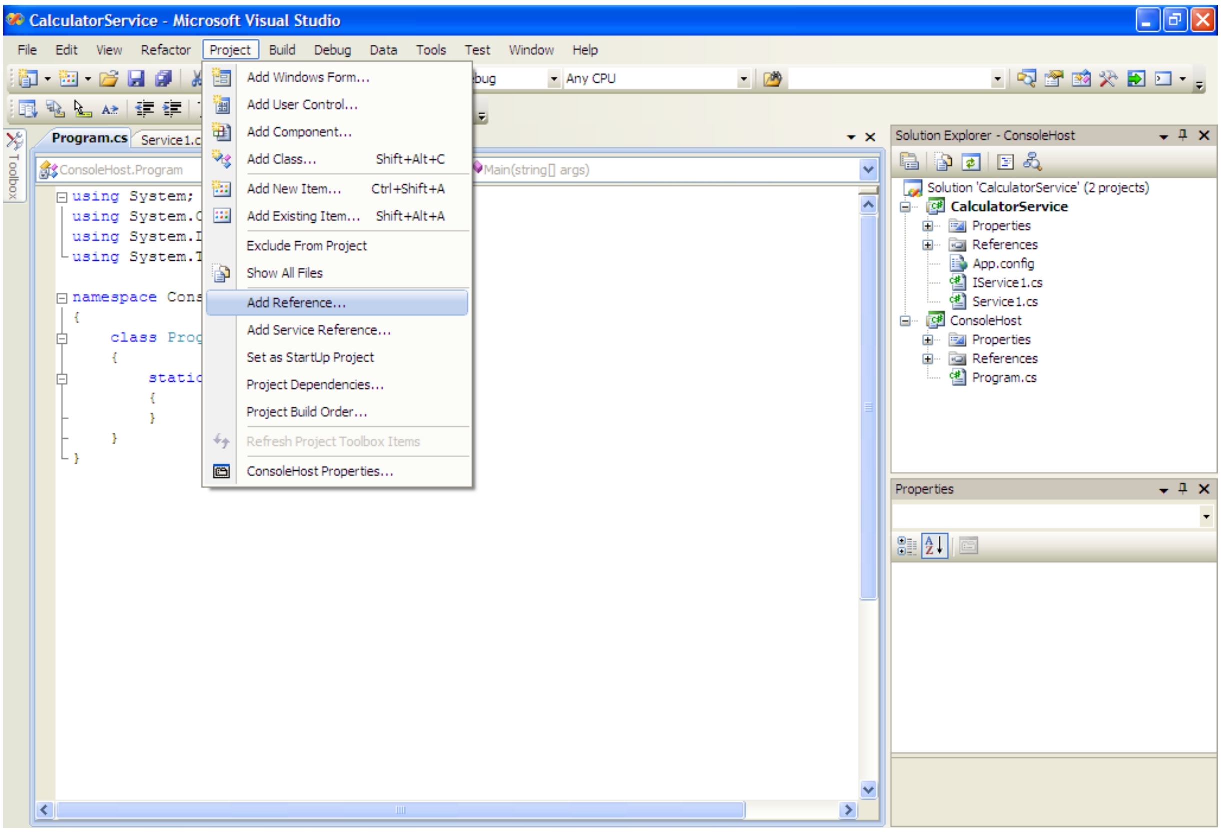


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace CalculatorService
{
    // NOTE: If you change the class name "Service1" here, you must also u
    public class Service1 : IService1
    {

        #region IService1 Members

        public int sum(int x, int y)
        {
            return (x + y);
        }

        public int diff(int x, int y)
        {
            return (x - y);
        }

        public int mul(int x, int y)
        {
            return (x * y);
        }
```

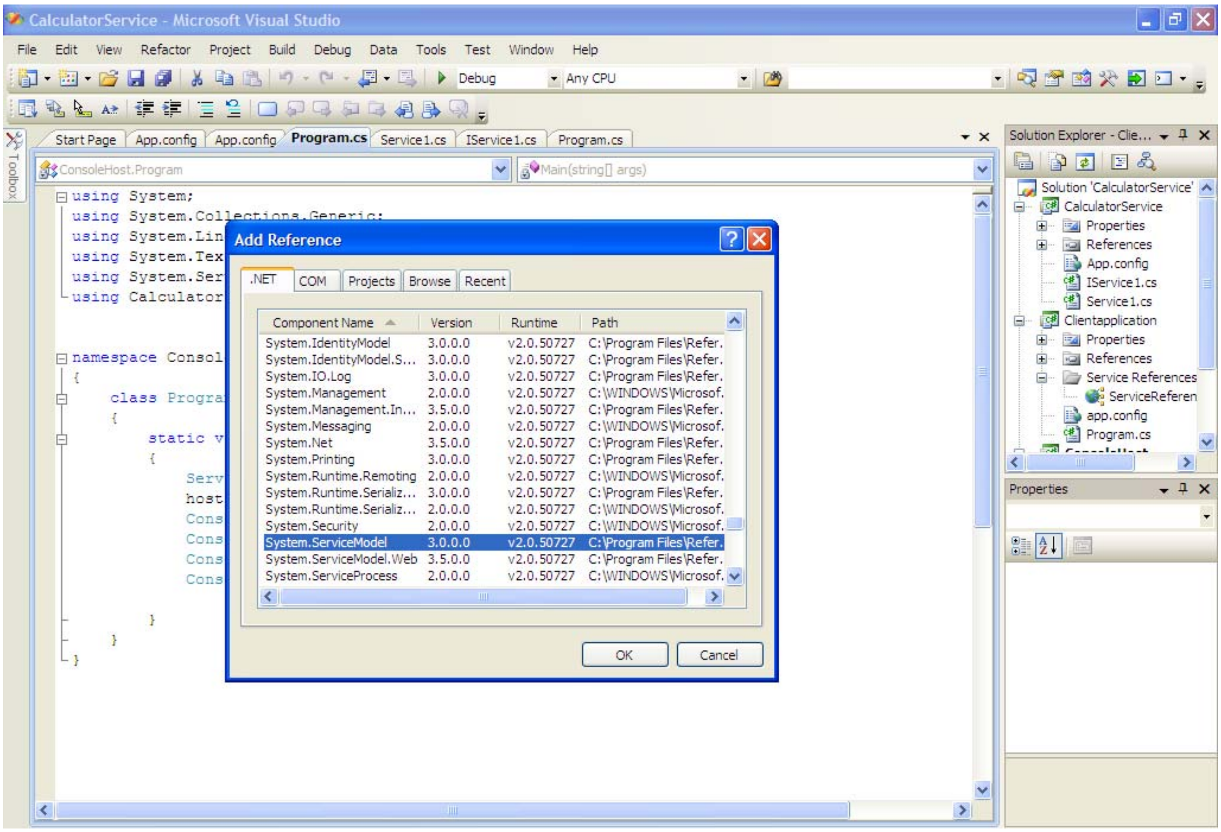
(Name)	CalculatorService
Active config	Debug|Any CPU
Description	
Path	C:\Documents and Settir
Startup project	CalculatorService

(Name)
The name of the solution file.

2. **Add the namespace ServiceModel and project CalculatorService by clicking on the project tab and then on add reference .Include it also in the ConsoleHost application as shown below :**



```
using System;
using System.C
using System.I
using System.T

namespace Cons
{
    class Prog
    {
        static
        {
        }
    }
}
```



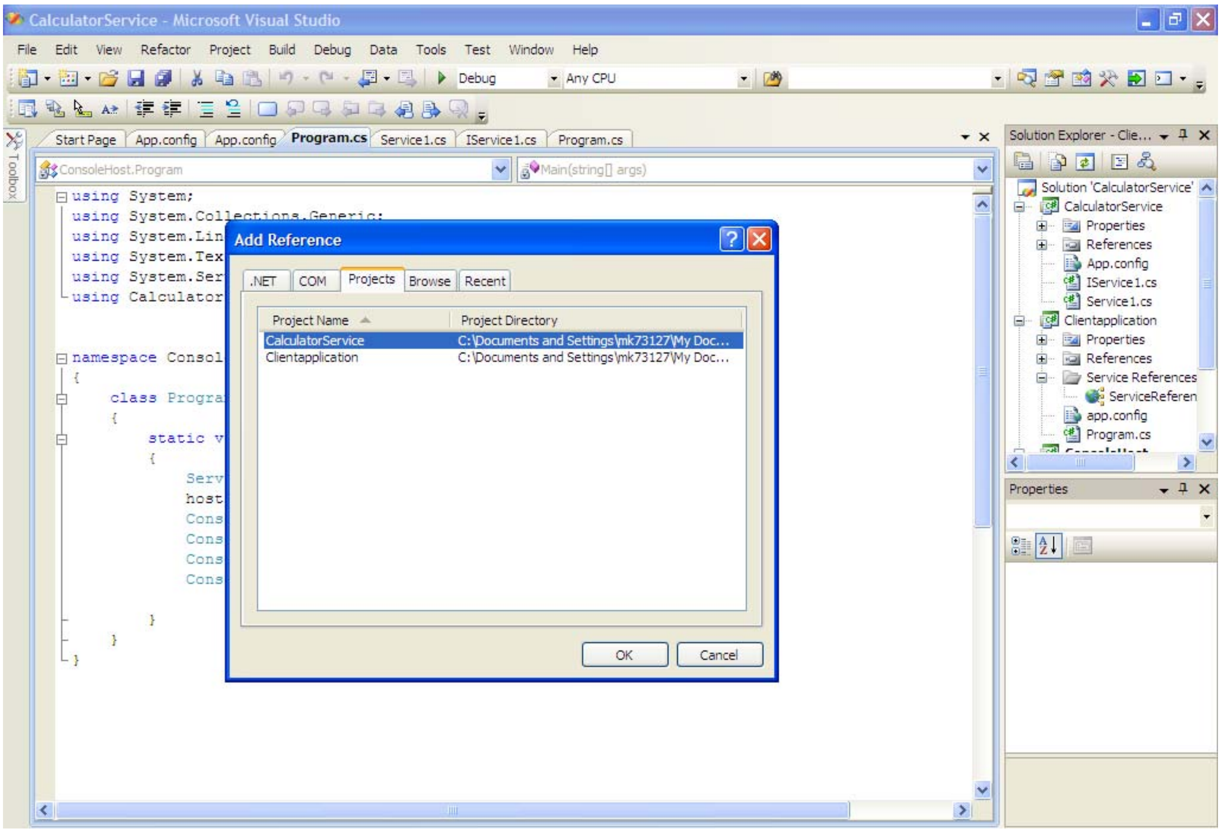
```
using System;
using System.Collections.Generic;
using System.Lin
using System.Tex
using System.Ser
using Calculator


namespace Consol
{
    class Progra
    {
        static v
        {
            Serv
            host
            Cons
            Cons
            Cons
            Cons

        }
    }
}
```

Add Reference

.NET | COM | Projects | Browse | Recent

Component Name	Version	Runtime	Path
System.IdentityModel	3.0.0.0	v2.0.50727	C:\Program Files\Refer.
System.IdentityModel.S...	3.0.0.0	v2.0.50727	C:\Program Files\Refer.
System.IO.Log	3.0.0.0	v2.0.50727	C:\Program Files\Refer.
System.Management	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Management.In...	3.5.0.0	v2.0.50727	C:\Program Files\Refer.
System.Messaging	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Net	3.5.0.0	v2.0.50727	C:\Program Files\Refer.
System.Printing	3.0.0.0	v2.0.50727	C:\Program Files\Refer.
System.Runtime.Remoting	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Runtime.Serializ...	3.0.0.0	v2.0.50727	C:\Program Files\Refer.
System.Runtime.Serializ...	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Security	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.ServiceModel	3.0.0.0	v2.0.50727	C:\Program Files\Refer.
System.ServiceModel.Web	3.5.0.0	v2.0.50727	C:\Program Files\Refer.
System.ServiceProcess	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.

OK | Cancel



```
using System;
using System.Collections.Generic;
using System.Lin
using System.Tex
using System.Ser
using Calculator

namespace Consol
{
    class Progra
    {
        static v
        {
            Serv
            host
            Cons
            Cons
            Cons
            Cons

        }
    }
}
```

Project Name	Project Directory
CalculatorService	C:\Documents and Settings\mk73127\My Doc...
Clientapplication	C:\Documents and Settings\mk73127\My Doc...

3. **Include the code for the ConsoleHost application given below**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.ServiceModel;
using CalculatorService;

namespace ConsoleHost
{
    class Program    {
        static void Main(string[] args)
        {
            ServiceHost host = new ServiceHost(typeof(Service1));
            host.Open();
            Console.WriteLine("WELCOME TO WCF WORLD");
            Console.WriteLine("--------------------------------------");
            Console.WriteLine("SERVICE IS READY TO WORK");
             Console.WriteLine("PRESS ENTER KEY TO STOP THE SERVICE…");
             Console.ReadLine();
             host.Close();
        }
    }
}
```

- Since the class ConsoleHost includes a Main method, it will run as a distinct process. To allow a WCF service to process requests from its clients, the process that hosts it must remain running.
- With WAS-hosted services, the standard process ensures the host remains running, but a hosting application must solve this problem on its own.
- In this simple example, the process is kept running through the straightforward mechanism of waiting for input from a console user.
- In a more realistic case, a WCF service hosted in this way would be running in a Windows service, allowing it to be started when a system boots, or be hosted in a GUI application, such as one built with Windows Presentation Foundation.

Defining Endpoints:

- Along with defining operations in a WCF service class and specifying a host process to run those operations, a WCF service must also expose one or more endpoints.
- Every endpoint specifies the following three things:
  - An address indicating where this endpoint can be found. Addresses are URLs that identify a machine and a particular endpoint on that machine.
  - A binding determining how this endpoint can be accessed. The binding determines what protocol combination can be used to access this endpoint along with other things, such as whether the communication is reliable.
  - A contract name indicating which service contract this WCF service class exposes via this endpoint.
- An easy way to remember what's required for WCF communication is to think of the ABCs of endpoints: address, binding, contract.
- Addresses are simple to understand—they're just URLs—and contracts have already been described.
- Bindings are also a critical part of how communication is accomplished, and they're worth further explanation.
- Suppose, for instance, that a service's creator wishes to allow clients to access that WCF service using either SOAP over HTTP or SOAP over TCP.
- Each of these is a distinct binding, so the WCF service would need to expose two endpoints, one with SOAP-over-HTTP binding and the other with a SOAP-over-TCP binding.
- To make bindings easier to use, WCF includes a set of predefined bindings, each of which specifies a particular group of options. Developers can configure these standard bindings if necessary, and they can also create wholly new custom bindings that provide exactly what a particular situation requires. Still, most applications will use one or more of the standard bindings that WCF provides
- Even though defining endpoints programmatically is possible, the most common approach today is to use a configuration file associated with the WCF service.
- Endpoint definitions embedded in code are difficult to change when a WCF service is deployed, yet some endpoint characteristics, such as the address, are very likely to differ in different deployments.
- Defining endpoints in configuration files (*.config) makes them easier to change, since changes don't require modifying and recompiling the source code for the WCF service class.
- For WCF services hosted in IIS or WAS, endpoints can be defined in the web.config file, while those hosted independently use the configuration file associated with the application they're running in (commonly referred to as app.config, although the actual filename varies). If used solely for the first CalculatorService service class shown earlier, this configuration file might look like this:

1. **Add an App.config file to ConsoleHost project and include in it the code given below**

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <services>
      <service name="CalculatorService.Service1">
        <endpoint
address="http://localhost:8731/Design_Time_Addresses/CalculatorService/Service1/mex"
binding="basicHttpBinding" contract="CalculatorService.IService1">
       </endpoint>
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

Practice Session:

- Host WCF Service created above using IIS

Lessons Learnt:

☑ Building Host Application

Topic: Consuming WCF Service

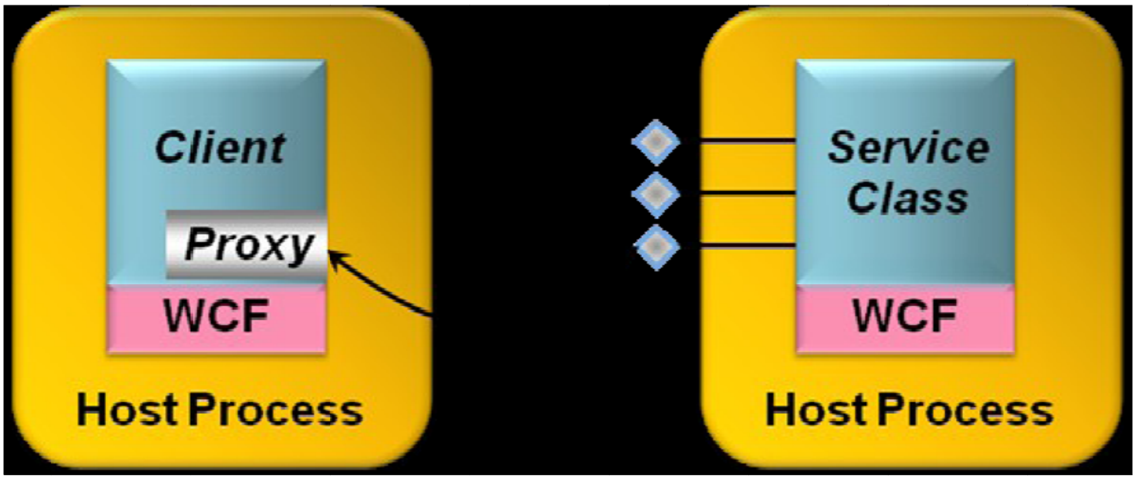
Estimated Time: 30 minutes.

Objectives: This module will familiarize the participant with

- Building a client application which consumes hosted WCF Service

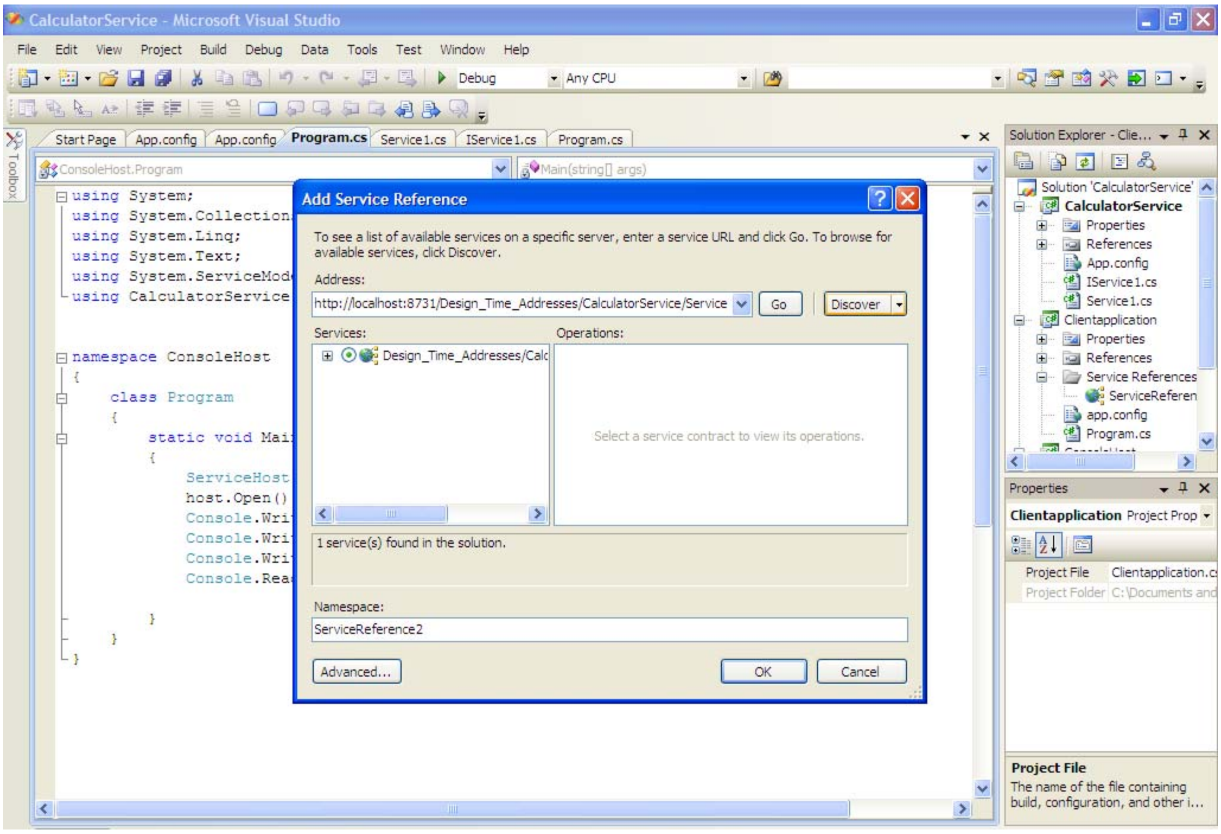
Presentation:

- Creating a basic WCF service isn't especially complicated.
- Creating a WCF client is even more straightforward. In the simplest approach, all that's required is to create a local stand-in for the service, called a proxy, that's connected to a particular endpoint on the target service, and then invoke the service's operations via the proxy. The figure below shows how this looks.



Demonstration/Code Snippet:

1. Right click on the solution explorer of the existing CalculatorService project, add new project and select console application and name it as ClientApplication.
2. Right click the ClientApplication project go to add service reference and enter the URL of the WCF service and click go.



```
using System;
using System.Collection
using System.Linq;
using System.Text;
using System.ServiceMod
using CalculatorService

namespace ConsoleHost
{
    class Program
    {
        static void Mai
        {
            ServiceHost
            host.Open()
            Console.Wri
            Console.Wri
            Console.Wri
            Console.Rea

        }
    }
}
```

Add Service Reference

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address: http://localhost:8731/Design_Time_Addresses/CalculatorService/Service

Services: Design_Time_Addresses/Calc

Operations: Select a service contract to view its operations.

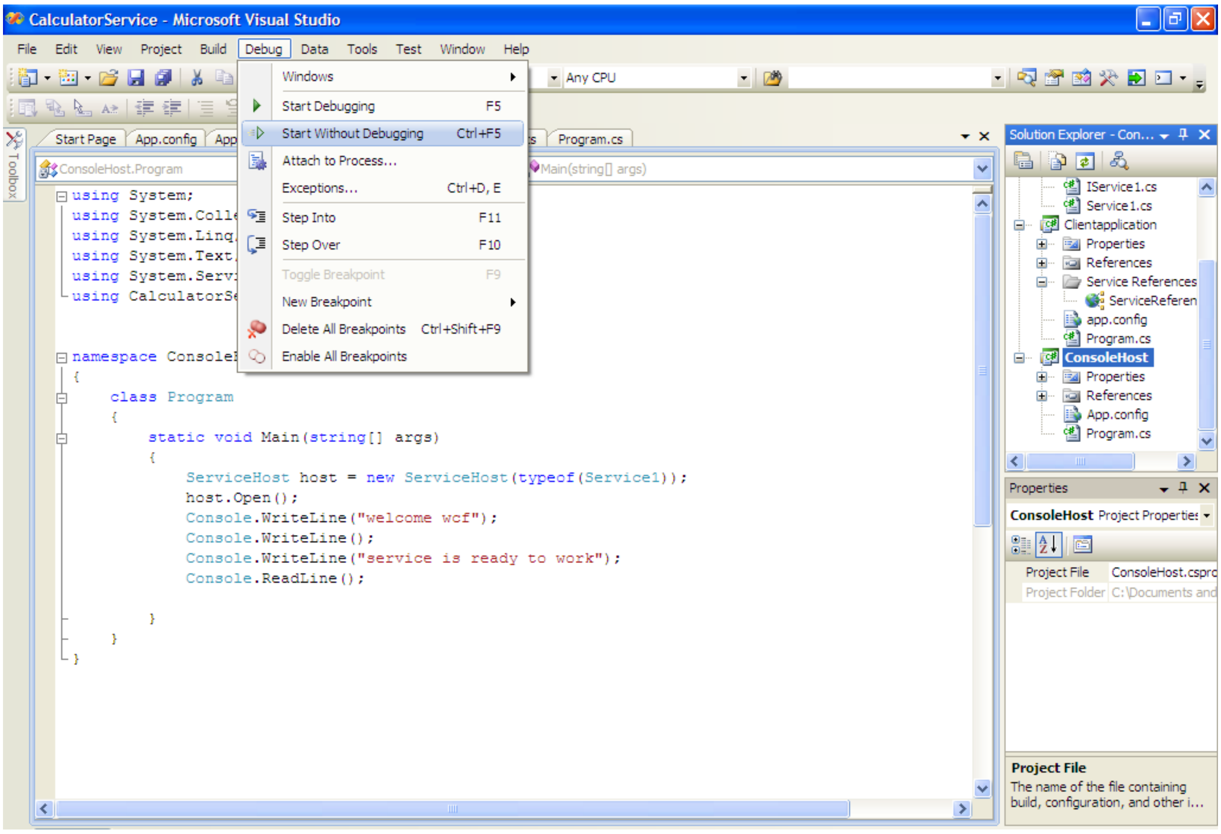
1 service(s) found in the solution.

Namespace: ServiceReference2

3. Add the service reference.
4. Include the namespace Clientapplication.ServiceReference1
5. Write the code for the ClientApplication as shown below

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Clientapplication.ServiceReference1;

namespace Clientapplication
{
    class Program
    {
        static void Main(string[] args)
        {
            int x, y, sum, diff, mul, div;
            Service1Client obj = new Service1Client();
            Console.WriteLine("enter the first number");
            x = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("enter the second number");
            y = Convert.ToInt32(Console.ReadLine());
            sum = obj.sum(x, y);
            diff = obj.diff(x, y);
            mul = obj.mul(x, y);
            div = obj.div(x, y);
            Console.WriteLine("The sum of two numbers are :");
            Console.WriteLine(sum);
            Console.WriteLine("The difference of two numbers are :");
            Console.WriteLine(diff);
            Console.WriteLine("The product of two numbera are :");
            Console.WriteLine(mul);
            Console.WriteLine("The division of two numbers are :");
            Console.WriteLine(div);
            Console.ReadLine();
        }
    }
}
```

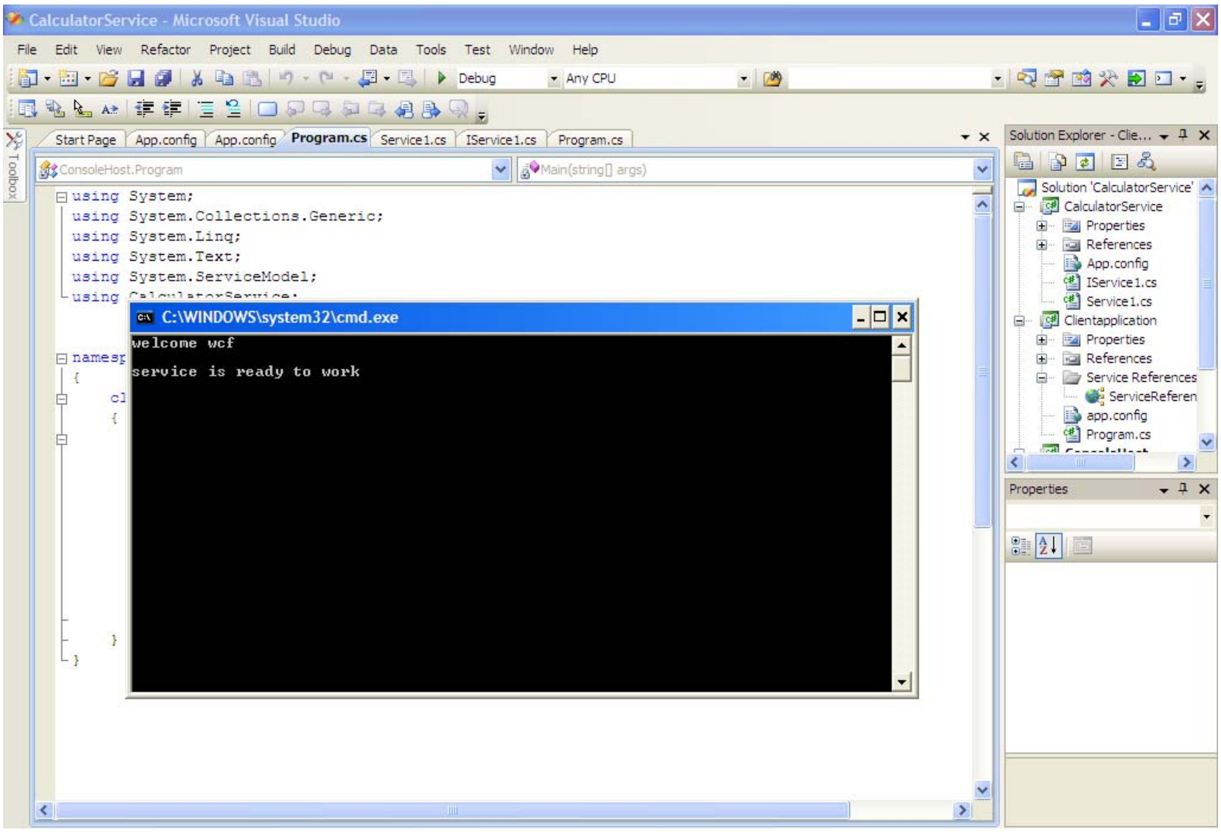
6. After code is being written right click the ConsoleHost project and make it as startup project
7. Click on the debug and click start without debugging option. Host application will run and output console window will appear as shown below



ConsoleHost.cspro

C:\Documents and

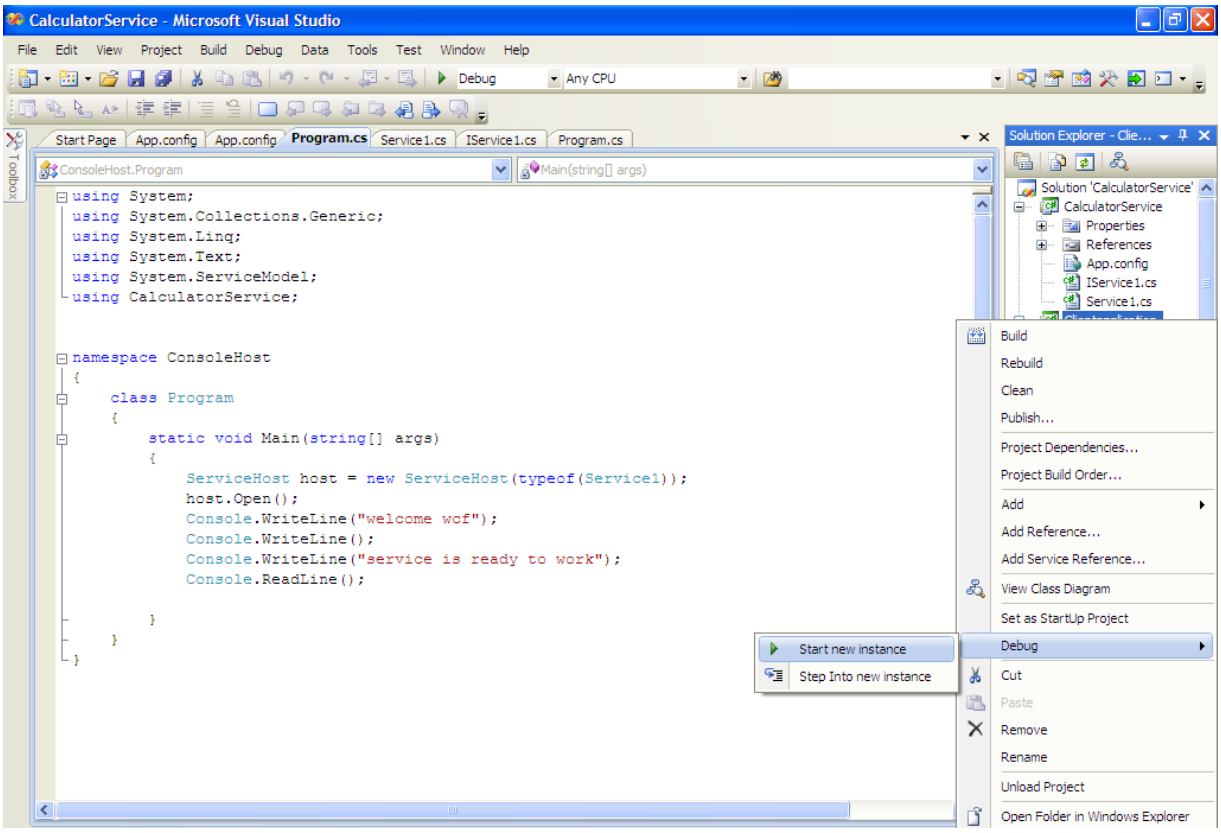
```
using System;
using System.Colle
using System.Linq
using System.Text
using System.Serv
using CalculatorSe

namespace Console
{
    class Program
    {
        static void Main(string[] args)
        {
            ServiceHost host = new ServiceHost(typeof(Service1));
            host.Open();
            Console.WriteLine("welcome wcf");
            Console.WriteLine();
            Console.WriteLine("service is ready to work");
            Console.ReadLine();

        }
    }
}
```



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ServiceModel;
using CalculatorService;
```

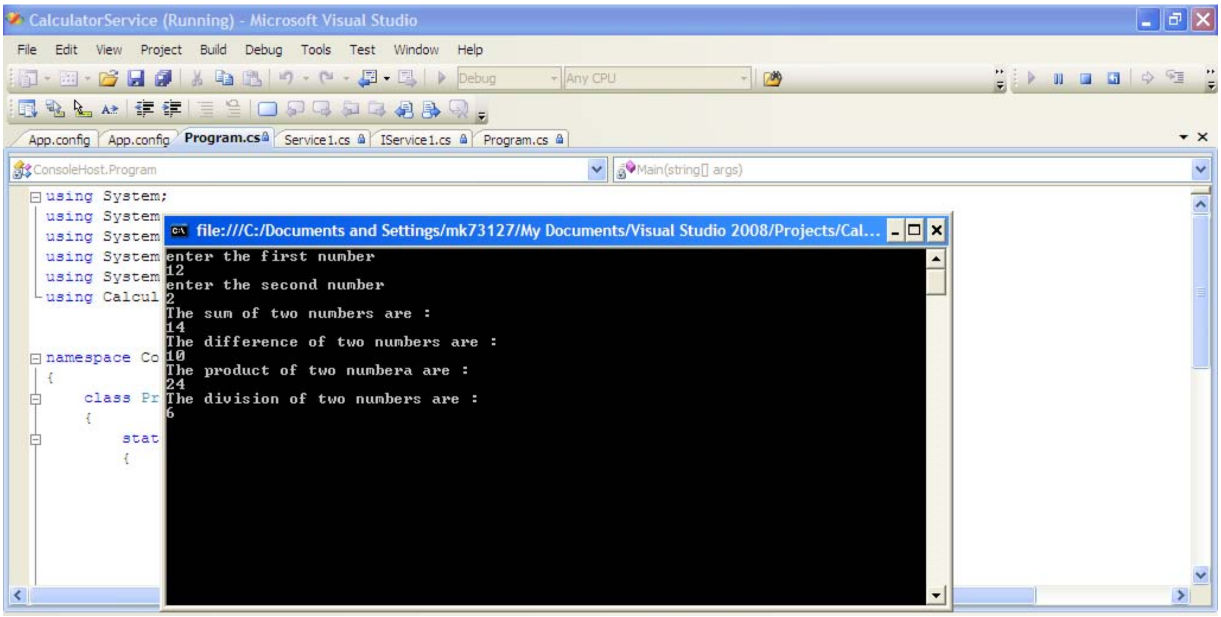
```
welcome wcf

service is ready to work
```

8. While the host application is running simultaneously runt the client application by right click on the Clientapplication project→Debug→Start new instance



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ServiceModel;
using CalculatorService;


namespace ConsoleHost
{
    class Program
    {
        static void Main(string[] args)
        {
            ServiceHost host = new ServiceHost(typeof(Service1));
            host.Open();
            Console.WriteLine("welcome wcf");
            Console.WriteLine();
            Console.WriteLine("service is ready to work");
            Console.ReadLine();

        }
    }
}
```

9. Client console window will appear where you will be entering the values of x and y and it will display the result as shown below


CalculatorService (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Tools Test Window Help
Debug
Any CPU
App.config
App.config
Program.cs
Service1.cs
IService1.cs
Program.cs
ConsoleHost.Program
Main(string[] args)
file:///C:/Documents and Settings/mk73127/My Documents/Visual Studio 2008/Projects/Cal...

```
using System;
using System
using System
using System
using System
using Calcul

namespace Co
{
    class Pr
    {
        stat
        {
```

```
enter the first number
12
enter the second number
2
The sum of two numbers are :
14
The difference of two numbers are :
10
The product of two numbera are :
24
The division of two numbers are :
6
```

Notes:

- All communication with a Windows Communication Foundation (WCF) service occurs through the endpoints of the service.
- The Windows Communication Foundation (WCF) can be thought of as a messaging infrastructure. WCF Service operations can receive messages, process them and send them messages. Messages are described using operation contract
- The run-time components of WCF can be divided into two major parts: The channel stack and the service framework, with the Message class being the connection point.
- The channel stack is responsible for converting between a valid Message instance and some action that corresponds to the sending or receiving of message data.

Practice Session:

- Explore the various methods, properties and events listed in the Message class and also the members of System.ServiceModel namespace.

Common Errors:

- Not including System.ServiceModel namespace in source code files (*.cs files)
- Running the client application without running the host application of WCF Service.

Lessons Learnt:

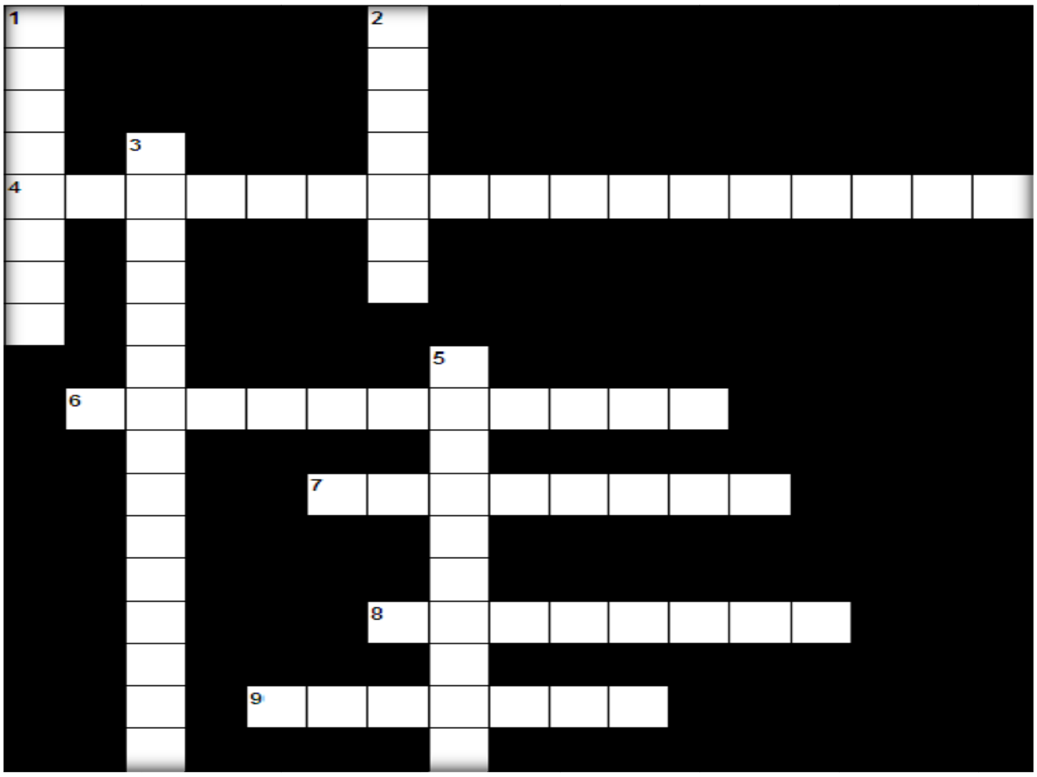
- ☑ Performance of Windows Communication Foundation (WCF) application depends on many different things.
- ☑ Application design, instancing mode, concurrency mode, transport, binding settings, and throttling settings all have effects on the performance of your application.
- ☑ Because WCF can communicate using Web services, interoperability with other platforms that also support SOAP, such as the leading J2EE-based application servers, is straightforward.
- ☑ Managing object lifetimes, defining distributed transactions, and other aspects of Enterprise Services are now provided by WCF.

Best Practices:

- Choosing the right size impacts how efficiently the network is put to use and the optimal capacity the service is running at.
- Use WCF because it provides a single, unified and extendable programming object model that can be used to interact with a number of previously diverse distributed technologies.

Crossword-3

Estimated Time: 10 minutes.



ACROSS:

4. The methods in WCF service class that can be invoked by client should be marked with ________ attribute. (18)
6. In order to do self hosting or host a WCF service in arbitrary process main method should create new instance of particular class. (11)
7. Are the core abstraction for sending and receiving messages from an endpoint (8)
8. ______ is a collection of operations that specifies what endpoint can communicate to outside world (8)
9. Specifies where to find WCF service (7)

DOWN:

1. It is single interface in which service can communicate with client. (8)
2. ____ specifies how client can communicate with endpoint including transport protocol, encoding and security requirements. (7)
3. Developer uses WCF defined _____ attribute to determine which of its methods are exposed as client callable operation. (15)
5. Are the types that modify or extend service or client functionality (9)

Answers for Crosswords

Crossword-1:

ACROSS	1)Service Contract 3) Endpoint 4) ServiceHost 7) Channels 8) Address 9) Binding
DOWN	2) OperationContract 5) Contract 6) Behaviors

Crossword-2:

ACROSS	2) REMOTING 3) XMLWEBSERVICES 5) WCF 7) DCOM 9) MSMQ
DOWN	1) WEBMETHOD 4) SOA 6) COM 8) COM+

Crossword-3:

ACROSS	4) OPERATIONCONTRACT 6) SERVICEHOST 7) CHANNELS 8) CONTRACT 9) ADDRESS
DOWN	1) ENDPOINT 2) BINDING 3) SERVICECONTRACT 5) BEHAVIORS

Team Members

Srikanth Nivarthi
47275

Sreenivas Ram
66223

Seshu Babu Barma
56150

Radhika Shashidhar jaji
75217

Ashok Sharma
62751

Veerendra Kumar Ankem
77964

Teena Arora
74572