WCF is a complex, yet robust and flexible service-oriented framework. By properly configuring WCF Services, developers can precisely expose business logic and data sources to clients in a variety of ways. WCF Services can send and receive messages as XML in a SOAP envelop, as well as RESTful formats, including POX (plain old XML), ATOM (an XML language used for web feeds), and JSON (JavaScript Object Notation).

**What is a Namespace?**

A *namespace name* is a [uniform resource identifier](http://en.wikipedia.org/wiki/Uniform_resource_identifier) (URI)

A **namespace** is used to declare a scope that contains a set of related objects. You can use a namespace to organize code elements and to create **globally unique types**.

**What is XML**

XML stands for EXtensible Markup Language.

XML was designed to describe data.

XML is a software- and hardware-independent tool for carrying information.

**What is an XML namespace?**

The namespace is defined by the **xmlns attribute** in the start tag of an element.

XML Namespaces provide a method to avoid element name conflicts.

XML namespaces are used for providing uniquely named [elements](http://en.wikipedia.org/wiki/Data_element) and attributes in an [XML](http://en.wikipedia.org/wiki/XML) document.

A simple example would be to consider an XML instance that contained references to a customer and an ordered product. Both the customer element and the product element could have a child element named **id**. References to the **id** element would therefore be ambiguous; placing them in different namespaces would remove the ambiguity.

**What is XSLT?**

XSLT (eXtensible Stylesheet Language Transformations)

## What is SOAP?

* SOAP stands for Simple Object Access Protocol
* SOAP is a communication protocol
* SOAP is for communication between applications
* SOAP is a format for sending messages
* SOAP communicates via Internet
* SOAP is platform independent
* SOAP is language independent
* SOAP is based on XML
* SOAP is simple and extensible
* **SOAP allows you to get around firewalls**
* SOAP is a W3C recommendation
* SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages. = Interoperable.

A SOAP message is an ordinary XML document containing the following elements:

* An Envelope element that identifies the XML document as a SOAP message
* A Header element that contains header information
* A Body element that contains call and response information
* A Fault element containing errors and status information

**What is a Session in WCF?**

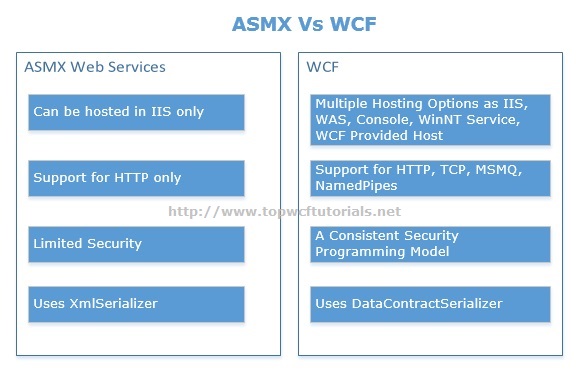
In WCF, there is always a service class instance that handles incoming service requests. These instances may already be there (at server when request arrives) or may be created as needed. In WCF, the concept of session is mainly to manage these service instances so that server can be utilized in an optimized way. At the server,

**What are the core components of WCF Service?**

A WCF service has at least following core components.

* Service Class:  A service class implementing in any CLR-based language and expose at least one method.
* Hosting Environment: a managed process for running service.
* Endpoint: a client uses it to communicate with service.

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

The basic difference is that ASMX web service is designed to send and receive messages using SOAP over HTTP only. While WCF service can exchange messages using any format (SOAP is default) over any transport protocol (HTTP, TCP/IP, MSMQ, Named Pipes etc).

**What are the Endpoints in WCF? or Explain ABCs of endpoint?**

For WCF services to be consumed, it’s necessary that it must be exposed; Clients need information about service to communicate with it. This is where service endpoints play their role.

A service endpoint has three basic elements or also called ABCs of an endpoint i.e. Address, Binding and Contract.

* ***Address****:*It defines “WHERE”. Address is the URL that identifies the location of the service.
* ***Binding****:*It defines “HOW”. Binding defines how the service can be accessed.
* ***Contract****:*It defines “WHAT”. Contract identifies what is exposed by the service.

### What is the difference between Service EndPoint and Client Endpoint?

Both Service and Client Endpoint has same ABC but we think in different perspective while working with them. We can differentiate between Service and Client Endpoint with respect to Address, Binding and Contract as:

|  |  |
| --- | --- |
| Service Endpoint | Client Endpoint |
| **WHERE**: URL of hosted service. | **WHERE**: where to connect with hosted service. |
| **WHAT**: bindings being used. | **WHAT**: binding supported by service. |
| **CONTRACT**: Service Contract i.e. interfaces | **CONTRACT**: what to pass and expect while communicating with service. |

**What is a WCF Binding? How many different types of bindings available in WCF?**

Bindings in WCF actually defines how to communicate with the service. Binding specifies what communication protocol as well as encoding method will be used. Optionally, binding can specify other important factors like transactions, reliable sessions and security.

There are different built-in bindings available in WCF

* basicHttpBinding

wsHttpBinding

* netNamedPipeBinding
* netTcpBinding
* netPeerTcpBinding
* netmsmqBinding

**Can we have multiple endpoints for different binding types in order to serve different types of clients?**

Yes, we can have multiple endpoints for different binding types. For example, an endpoint with wsHttpBinding and another one with netTcpBinging.

**What are Contracts in WCF?**

A Contract is basically an agreement between the two parties i.e. Service and Client. In WCF, Contracts can be categorized as behavioral or structural.

1. *Behavioral Contracts* on a service.
   * *ServiceContract* attribute is used to mark a type as Service contract that contains operations.
   * *OperationContract* attributes is used to mark the operations that will be exposed.
   * *Fault Contract* defines what errors are raised by the service being exposed.
2. *Structural Contracts*
   * *DataContract*  attribute define types that will be moved between the parties.
   * *MessageContract* attribute define the structure of SOAP message.

**What Message Exchange Patterns supported by WCF?**

Windows Communication Foundation supports the following Message Exchange Patterns(MEPs):

* Request/Response
* One Way
* Duplex

**Request/Response**  
It’s the default pattern. In this pattern, a response message will always be generated to consumer when the operation is called, even with the void return type. In this scenario, response will have empty SOAP body.

**One Way**  
In some cases, we are interested to send a message to service in order to execute certain business functionality but not interested in receiving anything back.

**Duplex**  
The Duplex MEP is basically a two-way message channel. In some cases, we want to send a message to initiate some longer-running process and require a notification back from service in order to confirm that the requested process has been completed. Both Request Reply or One-Way is available. **REQUEST REPLY IS SYNCRONOUSE AND ONE WAY WILL MIMIC ASYNCRONOUS**

### What are the different ways to expose WCF Metadata?

### By default, WCF doesn’t expose metadata. We can expose it by choosing one of the following ways: 1.    In configuration file, by enabling metadata exchange as follows:

### 2. Self Service Host can expose a metadata exchange endpoint by configuring MEX or WSDL in code.

### What are the differences between MEX and WDDL?

### MEX is Based on SOAP messages instead of http get requests

### The response that you will receive when calling a MEX endpoint’s GetMetadata operation will include the content of the WSDL and all the XSD files that are linked to it.

### MEX is designed to support non-HTTP protocols and for advanced configuration/security scenarios. WSDL is the legacy way and MEX is the new improved version for WCF.

### A WSDL is generally exposed through http or https get urls that you can't really configure (say for security limitations or for backward compatibility). MEX endpoints expose metadata over configurable endpoints, and can use different types of transports, such as TCP or HTTP, and different types of security mechanisms.

### So MEX are more configurable, while WSDL is more interoperable with older versions of clients and non-.net clients that work with WSDLs.

**what is the meaning of httpGetEnabled="true" or httpGetEnabled="true"?**

Enables service to expose metadata via WSDL through the default URI, even if you have not defined a MEX endpoint for your service.

**What is** **Web Services Description Language** (**WSDL**)?

WSDL is an XML-based language used to describe the services a business offers and to provide a way for individuals and other businesses to access those services electronically

**What is mexHttpBinding in WCF?**

In order to generate proxy, we need service metadata and *mexHttpBinding* is the binding that returns service metadata.

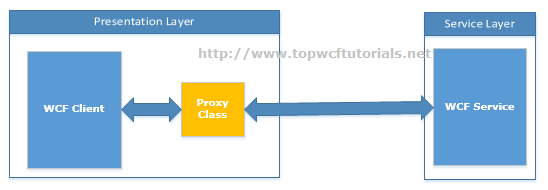
If we look into our configuration file, service will have an endpoint with mexHttpBinding as follows:mexHttpBinding

and service metadata behavior will be configured as follows:httpGetEnabled

Before deployment of application to production machine, it should be disabled.  
In order to support other protocols, related bindings are:

* mexHttpBinding
* mexHttpsBinding
* mexTcpBinding

**What is a Service Proxy in Windows Communication Foundation?**

A service proxy or simply proxy in WCF enables application(s) to interact with WCF Service by sending and receiving messages. It’s basically a class that encapsulates service details i.e. service path, service implementation technology, platform and communication protocol etc. It contains all the methods of service contract (signature only, not the implementation). So, when the application interact the service through proxy, it gives the impression that it’s communicating a local object.  
We can create proxy for a service by using Visual Studio or SvcUtil.exe.

**What are the different ways to generate proxy in WCF?**

Generating proxy using Visual Studio is simple and straight forward.

* Right click References and choose “Add Service Reference”.
* Provide base address of the service on “Add Service Reference” dialog box and click “Go” button. Service will be listed below.
* Provide namespace and click OK.

Visual studio will generate a proxy automatically.

We can generate proxy using svcutil.exe utility using command line. This utility requires few parameters like HTTP-GET address or the metadata exchange endpoint address and a proxy filename i.e. optional.

**Difference between using ChannelFactory and Proxies in WCF?**

A ChannelFactory creates a kind of Channel used by clients to communicate with service endpoints. If we have control over Server and Client, then ChannelFactory is a good option because it relies on having local interfaces that actually describes the service i.e. service contract.

On the other hand, If we don’t have control over server and only have WSDL/URL, then it’s better to generate proxy using Visual Studio or SvcUtil. SvcUtil is better option as compared to Visual Studio because we have more control in case of SvcUtil.

**How to create proxy for Non-WCF Services?**

In case of Non-WCF Services, we can create proxy by either using Visual Studio or svcUtil.exe tool by pointing to WSDL of the non-WCF service. In this scenario, we can’t create proxy through ChannelFactory or manually developing proxy class because we don’t have local interfaces i.e. service contract.

**Breifly explain Automatic Activation in WCF?**

Automatic activation means service starts and serves the request when a message request is received, but service doesn’t need to be running in advance. There are few scenarios in which service needs to be running in advance, For example, in case of Self-Hosting.

**What are the different WCF Instance Activation Methods available?**

WCF supports three different types of Instance Activation methods:

* **Per Call**: A new instance is created against each incoming request from client and later disposed off  as response generated
* **Per Session**: an instance for each session.
* **Singleton**: All incoming requests are served by only one instance.

**What are the different ways to handle concurrency in WCF?**

There are three different ways to handle concurrency in WCF that are:

* Single
* Multiple
* Reentrant

**Single**: means at a given time, only a single request can be processed by WCF service instance. Other requests will be waiting until the first one is fully served.  
**Multiple**: means multiple requests can be served by multiple threads of a single WCF service instance.  
**Reentrant**: means a single WCF service instance can process one request at a given time but the thread can exit the service to call another service.  
We can apply these concurrency settings by putting ConcurrencyMode property in ServiceBehavior as follows:

*[ServiceBehavior(****ConcurrencyMode = ConcurrencyMode.Multiple****] public class MyService : IMyService  
{  
}*

**What is WCF throttling?**

WCF throttling enables us to regulate the maximum number of WCF instances, concurrent calls and concurrent sessions. Basic purpose is to control our WCF service performance by using Service throttling behavior.

In configuration file we can set this behavior as follows:

*<serviceBehavior>*  
*<behavior name=”MyServiceBehavior”>*  
*<serviceThrottling*  
*maxConcurrentInstances=”2147483647”*  
*maxConcurrentCalls=”16″*  
*maxConcurrentSessions=”10″*  
*</behavior>*  
*</serviceBehavior>*

**What is a fault contract?**

Normally, by default, when some exception occurs at a WCF service level, it will not expose as it is to client. Reason is that WCF exception is a CLR exception and it doesn’t make sense to expose it outside CLR because it contains internal details of service code like stack trace. So, WCF handles and returns error details to client using Fault Contract.“*So, fault contract is a contract that contains the details of possible exception(s) that might occur in a service code.*”

**A user has a service with a one-way operation that includes a fault contract, and he gets an exception when he tries to host the service. Why?**

This is true, because, to return faults, the service requires some form of a two-way communication channel, which is not there with one-way operations.

**What are the core security concepts supported by WCF?**

There are four core security Features

* **Confidentiality**: It’s a confirmation about the recipient. Only the valid recipient can read the message when it passed between service and client.
* **Integrity**: is to ensure that message received has not been tempered with or changed during exchange.
* **Authentication**: is a way for the parties (sender and receiver) to identify each other.
* **Authorization**: ensures what actions and recourses an authenticated user can perform or access?

**Difference between Message Level security and Transport Level security?**

Security can be configured at two different levels in Windows Communication Foundation:

**Transport Level Security**secures the transport (the pipe) over which the message passes through from client to a service.

Transport Security is applied at the transport byte stream below the message layer. The message does not have a Security header and the message does not carry any user authentication data. It is the least flexible in terms of WS- Security usage and it is highly dependent on the transport. It is the fastest in terms of performance.

**Message Level Security**  
secures the message that is being transported from one end to another.

Message Security is applied at the message layer and it is transport independent. It is a point to point security model with maximum flexibility in terms of having the message routed over different transports. WS-Security defines different ways to secure a message and the tokens that can be used. Message Security provides the maximum flexibility in terms of that as well. Message Security is slowest in terms of performance.

### Mixed-Mode Security

### is a hybrid between Transport and Message Security. The transport is encrypted and the message contains some user authentication tokens. If the token can provide a key (i.e., it is not a username/password token) then it will sign the timestamp in the security header. If the client token is a Asymmetric token then the 'To' header will be signed as well. It is faster than Message Security.

### Can you please explain which security mode supported by various WCF Bindings?

Following table illustrates in details about support for security mode in Windows Communication Foundation for various WCF Bindings.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| WCF Binding | None | Transport | Message | Mixed | Both |
| **BasicHttpBinding** | Default | Yes | Yes | Yes | No |
| **WSHttpBinding** | Yes | Yes | Default | Yes | No |
| **WSDualHttpBinding** | Yes | No | Default | Yes | No |
| **NetTcpBinding** | Yes | Default | Yes | Yes | No |
| **NetNamedPipeBinding** | Yes | Default | No | No | No |
| **NetMsmqBinding** | Yes | Default | Yes | No | Yes |

**Difference between BasicHttpBinding and WsHttpBinding w.r.t Security?**

WsHttpBinding supports advanced WS-\* specification, it has a lot more security options available. For example, It provides message-level security i.e. message is not sent in plain text. Also it supports for WS-Trust and WS-Secure conversation.  
While in case of BasicHttpBinding, it has fewer security options, or we can say, there is no security provided, by default. At transport level, it can provide confidentiality through SSL.

### Please explain authorization options supported in WCF?

Authorization as a core feature of security in WCF supports different authorization types.

* **Role-based authorization** is the most common authorization approach being used. In this approach, authenticated user has assigned roles and system checks and verifies that either a specific assigned role can perform the operation requested.
* **Identity-based authorization** approach basically provides support for identity model feature which is considered to be an extension of role-based authorization option. In this approach, service verifies client claims against authorization policies and accordingly grant s or denies access to operation or resource.  
  **Resource-based authorization** approach is a bit different because it’s applied on individual resources and secure those using windows access control lists (ACLs).

**What is Reliable Messaging in WCF?**

Reliable massaging ensures that messages send over a wire are delivered exactly once, at least once or at most once; even in the presence of component, system, or network failures. If required the messages can also be delivered in the same order as they were sent.

WS-ReliableMessaging protocol enable the developer to insist on reliably delivering messages in both directions. The mechanism is actually transport agnostic and allows messages to be flown reliably from client to service and replies from service to client.

**Where do we set up reliable messaging in config file?**

On the endpoint binding.

**What are Reliable Sessions in WCF?**

Reliable sessions actually ensure that the caller for messages will know about the lost message(s) but it can’t guarantee about the delivery of message(s).

There is a misconception about reliable sessions that it ensures the session will never expire or stays for a very long time. This we can achieve by using timeout for sessions.

**Briefly explain WCF RESTful services?**

RESTful services are those which follow the REST (Representational State Transfer) architectural style. As we know that WCF allows us to make calls and exchange messages using SOAP over a variety of protocols i.e. HTTP, TCP, NamedPipes and MSMQ etc. In a scenario, if we are using SOAP over HTTP, we are just utilizing HTTP as a transport. But Http is much more than just a transport. So When we talk about REST architectural style, it dictates that:

***Instead of using complex mechanisms like CORBA, RPC or SOAP for communication, simply HTTP should be used for making calls.***

RESTful architecture uses HTTP for all CRUD operations like (Read/CREATE/Update/Delete) using simple HTTP verbs like (GET, POST, PUT, and DELETE). It’s simple as well as lightweight.

**What is SOA (Service Oriented Architecture) and how WCF supports it?**

In software engineering, a Service-Oriented Architecture (SOA) is a set of principles and methodologies for designing and developing software in the form of interoperable services. These services are well-defined business functionalities that are built as software components (discrete pieces of code and/or data structures) that can be reused for different purposes. SOA design principles are used during the phases of systems development and integration.

A service-oriented architecture consists of a set of components that enable distributed resources (applications, database systems, etc.) to interact with each other in support of business processing.

The advantage of an SOA is that it enables common and shared interfaces to be defined and created for distributed resources. This eliminates the need to develop multiple proprietary point-to-point connections between resources, which reduces IT development and maintenance efforts and encourages service reuse.

**What is Enterprise Service Bus(ESB)**Another important development in the SOA environment is an enterprise service bus (ESB). An ESB is a message-based capability that facilitates interaction between distributed resources. An ESB is not required for SOA, but it does increase the power and flexibility of SOA usage.

The objective of an ESB is to route messages between resources in a reliable manner; that is, it guarantees message delivery. Message routing may be done synchronously or asynchronously between source and target systems. Messages may be also be transformed from a source format into a target format as they pass through the bus. Other facilities offered by an ESB include load balancing and failover.

**What is Transaction Propagation? And how WCF support it?**

Transaction propagation is the ability to propagate transaction across the boundaries of a single service. Or in other words, we can say that a service can participate in a transaction that is initiated by a client.  
In a SOA environment, transaction propagation becomes a key requirement. As we know that WCF supports SOA, so it provides support for transaction propagation as well.  
To enable transaction propagation, we need to set the value of *TransactionFlow*  property of the binding being used. This can be done programmatically as follows:

*WSHttpBinding bindingBeingUsed = new WSHttpBinding();*  
*bindingBeingUsed.TransactionFlow = “true”;*

 Or It can be done declaratively by updating configuration file as follows:

*<bindings>*  
*<wsHttpBinding>*  
*<binding name=”binding1”*  
*transactionFlow=”true” />*  
*</wsHttpBinding>*  
*</bindings>*

Default value for *TransactionFlow* property is “False”.

**Do all WCF bindings support for Transaction Propagation?**

No. Not all WCF bindings support transaction propagation. Only following list of bindings support for it.

* wsHttpBinding
* netTcpBinding
* netNamedPipeBinding
* wsDualHttpBinding
* wsFederationHttpBinding

**What are the various Transaction Flow Options available in WCF?**

If a service is configured for Transaction Propagation, WCF further supports various options for service methods to be part of any transaction initiated outside service boundaries.

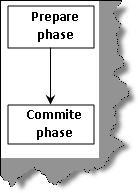
* **NotAllowed** Transaction Propagation is not allowed for particular service method. Its default value.
* **Allowed** Transaction Propagation is allowed but not compulsory.
* **Mandatory** Transaction Propagation is compulsory for that service method.

For example, Transaction Propagation is mandatory for CreditAccount service method in following code snippet.

*[ServiceContract]*  
*interface IPaymentService*  
*{*  
*[OperationContract]*  
*[TransactionFlow(TransactionFlowOption.Mandatory)]*  
*void CreditAccount(….);*  
*}*

**What is two-phase committed protocol?**

WCF transactions follow two phase commit. So there are two phases: the prepare phase and the commit phase. All co-ordination of transactions is done by the transaction manager.



In prepare phase, the transaction manager checks whether all entities are prepared to commit. In commit phase, the actual commit starts. You can think about the prepare phase as a check saying that all entities are ready to commit and in the commit

## Can you explain in depth how the prepare and commit phases work?

Let’s consider three computers as shown in the below figure. The client consuming the WCF service resides in computer A while computers B and C have the WCF services. The transaction is initiated from the computer A. So as we said previously, there are two phases: the prepare phase and the commit phase. In prepare phase, computer A sends messages to all WCF services saying, are they ready to commit? Once all WCF services respond saying that they are ready for commit, it starts the second phase. In the second phase, the WCF client issues a commit command. All the WCF services start execution and once they are done, they revert back saying they have committed. When all the WCF services revert saying they have committed, the transaction is marked as successful.

## Can we implement transactions in one-way WCF service calls?

No, In one-way WCF services, we do not get any response back, so WCF transactions are not possible with one way WCF service calls. WCF transactions are two-phase commits so every commit requires a response back from the service to the client

## Where does the transaction manager reside?

The transaction manager resides at the client computer which initiates the transaction.

using (TransactionScope ts = new TransactionScope(TransactionScopeOption.RequiresNew))

{

try

{

*// Call your webservice transactions here*

ts.Complete();

}

catch (Exception ex)

{

ts.Dispose();

}

}

**What are the supported transaction types in WCF?**

Supported transaction types in WCF are:

* Light Weight
* OLE Transactions
* WS-Atomic Transactions

**Simple way to enable Performance Counters supported by WCF is as follows:**

*<system.serviceModel>*  
*<diagnostics performanceCounters = “All” />*  
*</system.serviceModel>*

Above configuration setting will enable all categories of counters including ServiceModelService, ServiceModelEndpoint and ServiceModelOperation. Default value for it is “Off”.

**What is WCF WS-Atomic Transactions and What does ACID stand for?**

This is the ACID principle. ACID standing for Atomic, Consistent, Isolated, and Durable. Atomic means that the transaction is one whole unit of work. Consistent means that the results are predictable. Isolated means that this unit of work does not depend on some other work somewhere else. Durable means that if something goes wrong at any point, the transaction can be recovered.

**What process is started for each Application Pool when a request is made to a service in that pool?**

W3WP.exe (IIS Worker Process)

**aspNetCompatibilityEnabled?**

When this attribute is set to **true**, requests to Windows Communication Foundation (WCF) services flow through the ASP.NET HTTP pipeline, and communication over non-HTTP protocols is prohibited. The default is false.