**1**

**What is WCF?**

**WCF** stands for **Windows Communication Foundation**and is part of .NET 3.0. WCF is Microsoft platform for building distributed and interoperable applications.

**What is a distributed application?**  
In simple terms a distributed application, is an application where parts of it run on 2 or more computers. **Distributed applications**are also called as **connected systems.**

**What is an interoperable application?**  
An application that can communicate with any other application that is built on any platform and using any programming language is called as an interoperable application. Web services are interoperable, where as .NET remoting services are not.

**Why should we use WCF?**

Don't have to write ASMX, Remoting, TCP XML, Json applications to serve different customer needs. Just use multiple end points. unify and bring all these technologies under one roof Microsoft has come up with a single programming model that is called as WCF - Windows Communication Foundation.

**Ultimately, Windows Communication Foundation is a framework for building services that process XML messages. Windows Communication Foundation allows you to transmit messages using different transport protocols (such as HTTP, TCP, and MSMQ) and using different XML representations**

**Which specifications does WCF follow?**

WCF supports specifications defined by WS-\* specifications. WS-\* specifications are defined together by Microsoft, IBM, SUN and many other big companies so that they can expose there service through a common protocol. WCF supports all specifications defined we will understand them one by one.

**What is a WCF Proxy?**

A Proxy in Windows Communication Foundation is a class that enables client applications to communicate with a service by sending and receiving messages. It actually encapsulates a number of service details like service path, service implementation technology, platform being used, communication protocol etc. as well as all the methods

**How do you host multiple services in a single host?**

**you cannot!**

**either create one service class that implements both contract...**

**CustomerProcductService.cs implements both ICustomer and IProduct contracts.**

**GOLDEN RULE--- One SVC = One Service.**

**If self hosting same golden rule applies. One Service = One Service Class.**

**Other Option for self hosting is have two separate console apps**

**One for CustomerService.cs and One for ProductsService.cs**

---------------------------------------------------------------------------------------------------------

**2**

**Is it possible for a WCF service to implement multiple service contracts?**  
Yes, we make the service class implement multiple service interfaces, and then expose each service using a different endpoint.

**What is what is meant by Serialization** **in WCF?**

Serialization is the process of converting an object into an XML representation. The reverse process, that is reconstructing the same object from the XML is called as Deserialization.

**By default, WCF uses what serializer?**

**DataContractSerializer.**

With .NET 3.5 SP1 and above, we don't have to **explicitly use DataContract or DataMember**attributes. The **Data Contract Serializer**will serialize all public properties of your complex type in an alphabetical order. By default private field and properties are not serialized.  
  
If we decorate a complex type, with **[Serializable]** attribute the **DataContractSerializer**serializes all fields. With **[Serializable]**attribute we don't have explicit control on what fields to include and exclude in serialized data.  
  
If we decorate a complex type with **[Datacontract]**attribute, the **DataContractSerializer** serializes the fields marked with the **[DataMember]**attribute. The fields that are not marked with **[DataMember]**attribute are excluded from serialization. The **[DataMember]**attribute can be applied either on the **private fields**or **public properties.**  
  
In WCF, the most common way of serialization is to mark the type with the **DataContract**attribute and mark each member that needs to be serialized with the **DataMember**attribute.

If you want to have explicit control on what fields and properties get serialized then use DataContract and DataMember attributes.   
**1.** Using DataContractAttribute, you can define an XML namespace for your data  
**2.** Using DataMemberAttribute, you can  
    **a)** Define Name, Order, and whether if a property or field IsRequired  
    **b)** Also, serialize private fields and properties

**What Namespace does the Service Contract Attribute Live In?**

System.ServiceModel

**What Namespace does the Data and Data Member Contract Attributes Live In?**

System.Runtime.Serialization

------------------------------------------------------------------------------------------------

**3**

**If you expect the service to accept and return inherited types what attribute do you use?**

If we have classes related by inheritance, the wcf service generally accepts and returns the base type. If you expect the service to accept and return inherited types, then use KnownType attribute.

**There are 4 different ways to associate KnownTypes. What are they?**

**1.** Use **KnownType**attribute on the base type. This option is global, that is all service contracts and all operation contracts will respect the known types.

**2.** Apply **ServiceKnownType**attribute on the service contract. With this option the known types are respected by all operation contracts with in this service contract only.

**3.**If you want even more granular control, then apply **ServiceKnownType**attribute on specific operation contracts. With this option, only the operation contracts that are decorated with ServiceKnownType attribute respect known types.

**4.**You can also specify known types in the configuration file. This is equivalent to applying **KnownType**attribute on the base type, in the sense that it is applicable globally. All service contracts and operation contracts respect the known types.

---------------------------------------------------------------------------------------------------

**4**

**How do you enable tracing and message logging?**

Use **Microsoft Service Configuration Editor**to enable **tracing**and **message logging**in WCF. This can be done either on the client or the wcf service.

-------------------------------------------------------------------------------------------------

**5**

**Explain Difference between Data contract and Message Contract?**

With **Data Contracts**we have very limited control over the SOAP XML request and response messages that are generated. Use **Message Contracts,**if you want to have full control over the generated XML SOAP messages.

**MessageContract**gives full control over the SOAP messages by providing access to the SOAP header and body sections using **MessageHeader**and **MessageBodyMember**attributes. Use MessageContract if there is a reason to tweak the structure of the soap XML i.e if you want to include any additional information in the SOAP header.

**Few examples of when Message Contracts can be handy**  
**1.** Include some custom data in the SOAP header. In general SOAP headers are used to pass user credentials, license keys, session keys etc.  
**2.** Change the name of the wrapper element in the SOAP message or to remove it altogether.

----------------------------------------------------------------------------------------------------

**6**

**What happens when an exception occurs in a WCF service?  
What is a SOAP fault?  
How are WCF service exceptions reported to client applications?**

When an exception occurs in a WCF service, the service serializes the exception into a generic SOAP fault, and then sends the SOAP fault to the client. This generic soap fault contains no detail about the actual CLR exception.

By default unhandled exception details are not included in SOAP faults that are propagated to client applications for security reasons. Instead a generic SOAP fault is returned to the client.

For debugging use includeExceptionDetailInFaults="true"

A WCF service should be throwing a **FaultException**or **FaultException<T>** instead of **Dot Net exceptions.**This is because of the following 2 reasons.

**1.** An unhandled .NET exception will cause the channel between the client and the server to fault. Once the channel is in a faulted state we cannot use the client proxy anymore. We will have to re-create the proxy. We discussed this in [Part 17](http://csharp-video-tutorials.blogspot.com/2013/12/part-17-unhandled-exceptions-in-wcf_20.html) of the WCF tutorial. On the other hand faultexceptions will not cause the communication channel to fault.  
  
**2.** As .NET exceptions are platform specific, they can only be understood by a client that is also .NET. If you want the WCF service to be interoperable, then the service should be throwing FaultExceptions.

We can create strongly typed SOAP faults and throw them. Creating our own strongly typed SOAP faults allow us to include any additional custom information about the exception that has occurred.

In WCF, to centralize exception handling and to return a general faultreason to the client, we implement IErrorHandler interface. Let's now look at the 3 steps involved in centralizing exception handling in WCF.

### What is a fault contract?

### WCF handles and returns error details to client using Fault Contract.“So, fault contract is a contract that contains the details of exception that occur in service code.”

### CLR exceptions are not allowed to be passed through a WCF service channel.

### So we take the CLR exception details and prepare a Fault contract and respond back to client.

### http://www.topwcftutorials.net/wp-content/uploads/2012/05/WCF-Fault.jpg

-------------------------------------------------------------------------------------------------------------

**7**

**WCF service endpoint consists of 3 things What are they?**

**A - Address**(Address where the WCF Service is available)

**B - Binding** (Bindings specify how a WCF service endpoint communicates with other endpoints. A binding must specify the transport (for example, HTTP or TCP) to use. You can also set other characteristics, such as security and transaction support.)

**C - Contract** (Specifies what the service can do. For example, the service contract describes which operations the client can perform on the service)

**So, what is a binding in a WCF service?**  
WCF binding defines how the client needs to communicate with the service.

**The binding that you choose determines the following for the communication between the client and the service. What are They?**

**Transport Protocol** (for example, HTTP, TCP, NamedPipe, MSMQ)  
**Message Encoding** (for example, text/XML, binary, or (MTOM) Message Transmission Optimization Mechanism)  
**Protocols** (for example, reliable messaging, transaction support)

**In WCF there are several built-in bindings that we could use.  Name 5**

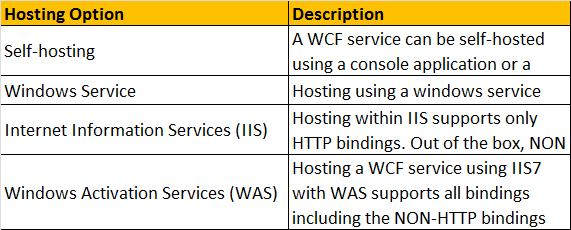
1. **BasicHttpBinding**: Basic web service communication. Exposes WCF services as legacy ASMX web services. Used for interoperability. No security by default.
2. **WSHttpBinding**: Web services with WS-\* support. Supports transactions and reliable messaging.
3. **WSDualHttpBinding**: Web services with duplex contract and transaction support.
4. **WSFederationHttpBinding**: Web services with federated security. Supports transactions.
5. **MsmqIntegrationBinding**: Communication directly with MSMQ applications. Supports transactions and queuing
6. **NetMsmqBinding**: Communication between WCF applications by using queuing. Supports transactions.
7. **NetNamedPipeBinding**: Communication between WCF applications on same computer. Supports duplex contracts and transactions.
8. **NetPeerTcpBinding**: Communication between computers across peer-to-peer services. Supports duplex contracts.
9. **NetTcpBinding**: Optimized communication between WCF applications across computers. Supports duplex contracts and transactions.

**We can configure an endpoint for WCF service in 2 ways What are they?**  
**1.** Declaratively using the configuration file  
**2.** Dynamically in code

**If dynamically, what namespaces would we use?**  
**1.** System.ServiceModel  
**2.** System.ServiceModel.Description

----------------------------------------------------------------------------------------------------------------------

**8**

**For a WCF service to be available for the clients to consume, we need to host the WCF service. What are the different WCF service hosting options?**   


In general, **self-hosting is only suitable during the development and demonstration phase**and not for hosting live WCF services.

**Advantages of self hosting a wcf service in a console application**  
**1.**Very easy to setup. Specify the configuration in app.config file and with a few lines of code we have the service up and running.  
**2.** Easy to debug as we don't have to attach a separate process that hosts the wcf service.  
**3.** Supports all bindings and transport protocols.  
**4.** Very flexible to control the lifetime of the services through the Open() and Close() methods of ServiceHost.

**What file is required to Host a WCF service in IIS?**

**.SVC File**

The Service Host directive in .svc file is responsible for creating an instance of Service Host when required. There is no need to write code to instantiate and start Service Host, as we did with self hosting.

**What WCF bindings are supported when hosting in IIS?**

Hosting WCF service in IIS 5.1 and IIS 6.0 is limited to HTTP communication only. This means we can only use HTTP related bindings.

**Windows Activation Service IIS7 allows hosting allows non-Http hosting**

**Windows Communication Foundation Non-HTTP Activation"**

**What are the Advantages of IIS Hosting**

**1. No code required to host the service:**The ServiceHost directive in .svc file is responsible for creating an instance of ServiceHost when required. There is no need to write code to instantiate and start ServiceHost, as is the case with self hosting.  
  
**2. Automatic message based activation:**IIS provides automatic message based activation. This means that the service can be activated on demand. When a message arrives at the service, it then launches itself and fulfils the request. In case of self hosting, the service should always be running.  
  
**3. Automatic process recycling:**IIS provides the capability of automatic process recycling, if the process is not healthy and if it's taking a long time to service the requests. We don't get automatic process recycling with self hosting.

**In IIS 7 how can we support binding other then the default HTTP protocol?**

 We need to install **"Windows Communication Foundation Non-HTTP Activation"** component and **WAS** **(Windows Process Activation Service)**component.

-------------------------------------------------------------------------------------------------------------------------------------------

**9**

**What is Message Exchange Pattern?**

Describes how the client and the WCF service exchange messages.

**WCF supports 3 Message Exchange Patterns. What are they?**

**1.** Request-Reply (**This is the Default**)  
**2.** One-Way   
**3.** Duplex

**The Message Exchange parameter is applied to what part of the WCF service contract?**

**[**OperationContract**]**

**[**OperationContract**(**IsOneWay=false**)]**

These are equivalent because Request-Reply is the default message exchange attribute.

**Explain Request-Reply:**  
In a **Request-Reply pattern**, the client sends a message to the WCF service and then waits for the reply message, even if the service operation's return type is void. This is the default.

**Explain One-Way:**

In case of **One-Way** operation, only one message is exchanged between the client and the service. The client makes a call to the service method, but does not wait for a response message.

**Faults if any while processing the request do not get reported back to the client.**

**Are One-Way calls the same as asynchronous calls?**

No,  One-way calls can still **block** the client, if the number of messages waiting to be processed has exceeded the server queue limit.

**[**OperationContract**(**IsOneWay=true**)]**

**Explain Duplex:**

A duplex service can send messages back to the client endpoint, providing event-like behavior.

**Where do we specify a callback contract?**

on the [ServiceContract CallbackContract](https://msdn.microsoft.com/en-us/library/system.servicemodel.servicecontractattribute.callbackcontract(v=vs.110).aspx) attribute.

Duplex messaging patterncan be implemented using **Request-Reply**or **One-Way** operations.

**How do we invoke the callback interface contract from the service code?**

System.ServiceModel.OperationContext.Current.GetCallbackChannel

**NOTE:** Not all bindings support callback operations. Only bidirectional-capable bindings (TCP, IPC) support callback operations.  
For example, because of its connectionless nature, HTTP cannot be used for callbacks, and therefore you cannot use callbacks over the BasicHttpBinding or the WSHttpBinding.

The MSMQ bindings also don’t support duplex communication.

**What HTTP Binding does support callbacks**?

To support callbacks over HTTP, WCF offers the WSDualHttpBinding, which actually sets up two WS channels:  
one for the calls from the client to the service and one for the calls from the service to the client.

----------------------------------------------------------------------------------------------------------------------------------------

**10**

**The default message encoding mechanism in WCF?**

Base64 text encoding.

**What are 2 disadvantages of base64 encoded messages?**

**1.** Base64 encoding bloats the message size by approximately 33%.  
**2.** Involves additional processing overhead to encode and decode.

**What is the preferred approach to send large binary messages in WCF?**

MTOM Encoding

**What does MTOM stand for?**

Message Transmission Optimization Mechanism.

**What does MIME stand for?**

Multipurpose Internet Mail Extensions

**How is MTOM binary data sent in the SOAP message?**

With MTOM, binary data is included as a MIME (Multipurpose Internet Mail Extensions) attachment.

-------------------------------------------------------------------------------------------------------------

**11**

**There are 3 instancing modes, What are they?**

**1. PerCall -** A new instance of service object is created for every request, irrespective of whether the request comes from the same client or a different client.  
  
**2. PerSession -**A new instance of the service object is created for each new client session and maintained for the duration of that session.  
  
**3. Single -**A single instance of the service object is created and handles all requests for the lifetime of the application, irrespective of whether the request comes from the same client or a different client.

**How do you specify what instancing mode you want to use?**  
Use ServiceBehavior attribute and specify InstanceContextMode parameters **on the Service Class**

**What are implications with Single instance context mode?**  
**1.** Since a single service instance is serving all client requests, state is maintained and shared not only between requests from the same client but also between different clients.  
**2.** Concurrency is an issue  
**3.** Throughput can be an issue. To fix the concurrency issue, we can configure the service to allow only a single thread to access the service instance. But the moment we do it throughput becomes an issue as other requests queue up and wait for the current thread to finish it's work.

------------------------------------------------------------------------------------------------------

**12**

**What is the SessionMode enumeration do?**

Specifying this enumeration on the Service Contract will require, allow, or prohibit bindings to use sessions.

**SessionMode enum has three members, What are they?**

**Allowed :**Service contract supports sessions if the binding supports them. This is the default if we have not explicitly specified the SessionMode on Service contract.   
  
**NotAllowed :**Service contract does not support bindings that initiate sessions.  
  
**Required :**Service contract requires a binding that supports session.

**Does the basic HTTP binding support sessions?**

No

WS HTTP Binding **is a** *session-based* binding

-----------------------------------------------------------------------------------------------------

**13**

**WCF Threading with Concurrency Mode**

**What are the three concurrency mode parameters and on what attribute are they set?**

* **ConcurrencyMode** can be   
  **Single** Single threaded. subsequent calls are placed in a queue until the service lock is released. Can cause deadlock when calling other services from the service.
* **Reentrant** Single threaded but accepts reentrant/returning calls made from the service to other services. Helps to avoid deadlock situations.
* **Multiple** Multi threaded. Synchronization and consistency is the service own responsibility. May be hard to manage in code.
* **Single**: A single request has access to the WCF service object at a given moment oft time. So only one request will be processed at any given moment of time. The other requests have to wait until the request processed by the WCF service is completed.
* **Multiple**: In this scenario, multiple requests can be handled by the WCF service object at any given moment of time. In other words, requests are processed at the same time by spawning multiple threads on the WCF server object. So you have great throughput here but you need to ensure concurrency issues related to WCF server objects.
* **Reentrant**: A single request thread has access to the WCF service object, but the thread can exit the WCF service to call another WCF service or can also call a WCF client through callback and reenter without deadlock.

[ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Single)]

**What is the meaning of concurrency?**

Happening at the same time.

**What is the difference between Instancing and Concurrency?**

WCF instancing and WCF concurrency are two different things. WCF instance dictates how objects are created while WCF concurrency dictates how many requests can be handled by WCF objects.

---------------------------------------------------------------------------------------------------------

**14**

**There are three throttling parameter in the Service Throttling configuration entry**

**What are they?**

**Throttling settings in config file**

<serviceBehaviors>

<behavior name="mexServiceBehavior">

<serviceMetadata httpGetEnabled="true" />

<serviceThrottling

maxConcurrentCalls="3"

maxConcurrentInstances="2"

maxConcurrentSessions="100"/>

</behavior>

</serviceBehaviors

---------------------------------------------------------------------------------------------------------

**15**

**What are the four fundamental security terms? And what do they mean?**

**Authentication -**The process of identifying the sender and recipient of the message.  
  
**Authorization -**The process of determining what rights the authenticated user has.  
  
**Confidentiality -**The process of ensuring that only the intended recipient of the message can view the message as it is being transmitted from the sender to the receiver. We achieve confidentiality by encrypting the message.  
  
**Integrity -**The process of ensuring that the message is not tampered with by a malicious user as it is being transmitted from the sender to the receiver. We achieve Integrity by signing the messages.

**The Protection Level enum is present in what name space?**

System.Net.Security namespace.

**What is the the default security scheme for NetTcpBinding?**

Transport

**What is the the default security scheme for wsHttpBinding?**

Message

**Can wsHttpBinding use Transport Security?**

Yes

**What is Message Security?**

Securing the message itself by encapsulating the security credentials with every SOAP message is called message security. As the message itself is protected, it provides end to end security.

**Define Transport Security for HTTP?**

HTTP provides transport security by using Secure Sockets Layer (SSL) over HTTP. Transport security provides only point-to-point channel security.

**Define Transport Security for TCP (Transmission Control Protocol)**

The operating system security for TCP by implementing point to point Transport Layer Security (TLS)

**What does the contract protection level attribute enum do?**

It allows for more granular message protection control at the contract level.

**What are the three ProtectionLevel named parameters enum?**

**None -**No protection. Message is not signed and not encrypted. Provides only authentication.

**Sign -**No encryption but is digitally signed to ensure the integrity of the message

**EncryptAndSign -**Message is encrypted and then signed to ensure confidentiality and integrity of the message

**What contract attribute can these named parameters be specified on?**

**ServiceContractAttribute**  
**OperationContractAttribute**  
**FaultContractAttribute**  
**MessageContractAttribute**

**What namespace contains the ProtectionLevel enum?**

System.Net.Security

**With message based security are the messages decrypted when the message is received?**

No, It requires the client or service to understand Soap XML encryption in order to decrypt

**With transport based security are the messages decrypted when the message is received?**

Yes, That is because the message is signed and encrypted at the Transport Layer and is decrypted at the transport layer by the time the message is recieved.

**What is the Default ClientCredentialType for wsHttpBinding?**

Windows.

**With wsHttpBinding what do we have to do to enable Transport security Mode with a Basic Client Credential Type**

**What does this attribute do?**

**What namespace do you include to use this attribute?**

[AspNetCompatibilityRequirements(RequirementsMode =

AspNetCompatibilityRequirementsMode.Allowed)]

ASP.NET compatibility mode, lets WCF service works as a normal ASMX service and supports all existing ASP.NET features. By setting compatibility mode, the WCF service will need to be hosted on IIS and communicate with its client application using HTTP

System.ServiceModel.Activation.

**IMetadataExchange Interface?**

Exposes methods used to return metadata about a service. When programming Windows Communication Foundation (WCF) services, it is useful to publish metadata about the service. For example, metadata can be a Web Services Description Language (WSDL) document that describes all of the methods and data types employed by a service. Returning metadata about an WCF service allows consumers of a service to easily create clients for the service.

**How did you do caching in WCF services?**

Microsoft has now developed an in-process memory cache that does not rely on the ASP.NET framework.  
This cache can be found in the “System.Runtime.Caching.dll” assembly.  
This sample pizza service caches the results after the first IIS message activation of the service.  
The first call takes some time because the data must first be retrived and chached but after that the cached data is returned very quickly.

---------------------------------------------------------------------------------------------------------------

TRANSACTIONS?

**Impersonation?**

**Session State?**

**Memory Cache**

**Tips!**

* **Always create the service with Interface->Implementation format, mention the contract in Interface.**
* **Define the service in Class library and refer the class library in Host project. Don’t use service class in host project.**
* **Change the instance mode to per call as default.**
* **Always catch exception using try/catch block and throw exception using FaultException < T >.**
* **Logging and Include exception should be enable while compiling the project in debug mode. While in production deployment disable the logging and Include exception details.**