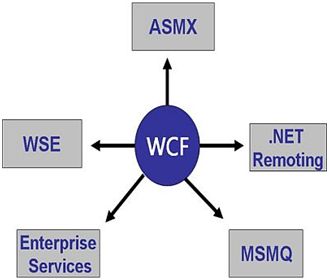
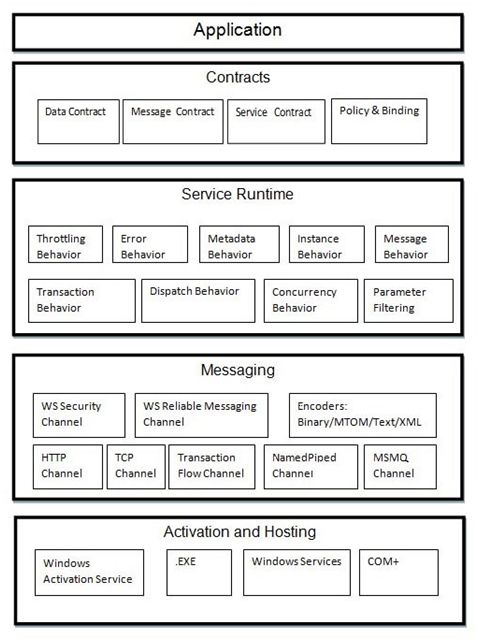
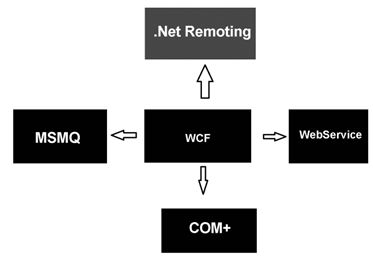
WCF Interview Questions and Answers

**Question 1: What is WCF?**

WCF is a platform for building distributed businesses and deploying services among various endpoints in Windows. WCF was initially called “**Indigo**” and we can build service-oriented applications and provide interoperability. WCF or Windows Communication Foundation is a programming model to create service oriented applications. It is used to create and deploy the service that is accessible to lots of different clients. It provides an environment where you can create a service which can be accessible to Windows clients as well as Linux clients or any others. It provides more features compared to web services. WCF is a Microsoft technology to create service oriented application. Before the WCF, the Web Service was used to create services but that type of service is only accessible to Windows client hosted on HTTP protocol. But WCF services are accessible with different protocols like http, tcp, msmq, etc. A few sample scenarios include:

  
  
**Question 2: Explain WCF Architecture and also explain its Fundamentals?  
Answer**  
Windows Communication Foundation (WCF) is a framework for building service-oriented applications by which we can send asynchronous message/data from one service endpoint to another service endpoint.  
WCF is a runtime and a set of APIs for creating systems that sends messages between services and clients. The same infrastructure and APIs are used to create applications that communicate with other applications on the same computer system or on a system that resides in another company and is accessed over the Internet.  
The WCF fundamentals are as follows:

* Unification
  + COM+ Services
  + Web Services
  + .NET Remoting
  + Microsoft Message Queuing
* Interoperability
* Service Orientation

**WCF Architecture**There are four major layers that provide developers with a new service-oriented programming model. The WCF architecture consists of the following layers.  
  
  
**Question 3: Why Should We Use WCF Service?  
Answer**  
Windows Communication Foundation (WCF) is a framework for building service-oriented applications. Most of WCF functionality is included in a single assembly called System.ServiceModel.dll, located in the System.ServiceModel namespace.   
  
  
  
**Why WCF**

1. A web service to exchange messages in XML format using HTTP protocol for interoperability.
2. A remoting service to exchange messages in binary format using TCP protocol for performance.
   * A secure service to process business transactions.
   * A service that supplies current data to others, such as a traffic report or other monitoring service.
   * A chat service that allows two people to communicate or exchange data in real time.
   * A dashboard application that polls one or more services for data and presents it in a logical presentation.
   * A Silverlight application to poll a service for the latest data feeds.

**Question 4: What are the features and advantage of WCF?  
Answer  
Features of WCF**  
Windows Communication Foundation (WCF) is a secure, reliable, and scalable messaging platform for the .NET Framework 3.0,

* Service Orientation
* Interoperability
* Multiple Message Patterns
* Service Metadata
* Data Contracts
* Security
* Multiple Transports and Encodings
* Reliable and Queued Messages
* Durable Messages
* Transactions
* AJAX and REST Support
* Extensibility

**Advantages of WCF**

1. Service Oriented
2. Location Independent
3. Language Independent
4. Platform Independent
5. Support Multiple operation
6. WCF can maintain transaction like COM+ Does
7. It can maintain state
8. It can control concurrency
9. It can be hosted on IIS, WAS, Self hosting, Windows services.

It has AJAX Integration and JSON (JavaScript object notation) support.

* WCF can be configured to work independently of SOAP and use RSS instead.
* WCF is one of the fastest communication technologies and offers excellent performance compared to other Microsoft specifications.
* To improve communication, transmission speed needs to be optimized. This is done by transmitting binary-coded XML data instead of plain text to decrease latency.
* Object life-cycle management and distributed transaction management are applicable to any application developed using WCF.

**Question 5: What is the difference between WCF and Web services?  
Answer: WCF**  
Windows Communication Foundation (WCF) is a framework for building service-oriented applications. Using WCF, you can send data as asynchronous messages from one service endpoint to another.  
**Web Services:** A Web Service is programmable application logic accessible via standard web protocols. One of these web protocols is the Simple Object Access Protocol (SOAP). SOAP is a W3C submitted note (as of May 2000) that uses standards based technologies (XML for data description and HTTP for transport) to encode and transmit application data.

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

**Question 6: Explain what is SOA? Are web-services SOA?  
Answer:**  **SOA**SOA stands for service oriented architecture. Service Oriented Architecture is an architectural approach in software development where the application is organized as "**Services**". Services are a group of methods that contain the business logic to connect a DB or other services. For instance you go to a hotel and order food. Your order first goes to the counter and then it goes to the kitchen where the food is prepared and finally the waiter serves the food.  
  
  
**Some important characteristics of Service Oriented Architecture**

* SOA services should be **independent** of other services.   
    
  Altering a service should not affect the client calling the service.
* Services should be **self-contained.**  
  • Services should be able to **define themselves** (in the Web Service Description Language (WSDL)). Services should be able to describe what they do. It should be able to tell the client what all of the operations it does, what are all the data types it uses and what kind of value it will return.

**SOA Overview:** A Service Oriented Architecture is based on four key abstractions.

* An Application Front End
* A Service
* A Service Repository
* A Service Bus

**Application Front End:** The application front end is decoupled from the services. Each service has a contract that defines what it will do and one or more interfaces to implement the contract.  
  
**A Service:** It has the methods with the defined contracts and the implementation of the business logic to connect the DB or other service.  
  
**A Service Repository:** The service repository provides a home for the services and the service bus provides an industry-standard mechanism for connecting to and interacting with the services.  
  
**Are web-services SOA?**  
  
SOA is thinking, it's an architectural concept and web service is one of the technical approach to complete it. Web services are the preferred standards to achieve SOA.

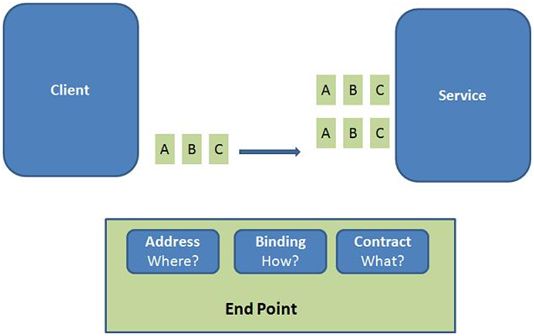
* In SOA we need the services to be loosely coupled. A web service communicates using SOAP protocol which is XML based which is very loosely coupled. SOA services should be able to describe themselves. WSDL describes how we can access the service.
* SOA services are located in a directory. UDDI describes where we can get the web service. This nothing but implementation of SOA registry.

**Question 7:What is Service Contract in WCF?  
Answer:** Service contract means the collective mechanisms by which a service's capabilities and requirements are specified for its consumers. We must say that it defines the operations that a service will perform when executed. It tells more things about a service, like message data types, operation locations, the protocols the client will need in order to communicate with the service.  
To create a service contract you define an interface with related methods representative of a collection of service operations, and then decorate the interface/class with the **ServiceContract** Attribute to indicate it is a service contract. Methods in the interface that should be included in the service contract are decorated with the **OperationContract** Attribute.   
**How to define a Service Contract  
  
**  
**ServiceContractAttribute:** It is used to declare the type as a Service Contract. It can be declared without any parameters but it can also take named parameters.

1. [ServiceContract(Name="MyService", Namespace="http://tempuri.org")]
2. public interface IMyService
3. {
4. [OperationContract]
5. **int** AddNum(**string** numdesc, **string** assignedTo);
6. }

**OperationContractAttribute:** It can only be applied on methods. It is used to declare methods which belong to a Service Contract. It controls the service description and message formats.  
  
For more details visit the following link:

* [Service Contract in WCF](http://www.c-sharpcorner.com/UploadFile/mahakgupta/service-contract-in-wcf/)

**Question 8: What are Contracts in WCF? How many Contracts are in WPF?**  
**Answer:** The main use of contracts is to allow the client and services agree as to the types of operations and structures they will use during the communication process. It also shows the formal agreements between client and service to define a platform-neutral and standard for describing that what the service does.  
  
  
  
**Service Contract:**A service contract defines the operations which are exposed by the service to the outside world. A service contract is the interface of the WCF service and it tells the outside world what the service can do. It may have service-level settings, such as the name of the service and namespace for the service.  
**Operation Contract:**An Operation Contract defines the method exposed to the client to exchange the information between the client and server. An Operation Contract describes what functionality is to be given to the client, such as addition, subtraction and so on.  
It can be defined as in the following:

1. **public** **interface** IService1
2. {
3. [OperationContract]
4. **string** GetData(**int** value);
5. [OperationContract]
6. CompositeType GetDataUsingDataContract(CompositeType composite);
7. }

**Data Contract:**Data Contracts define the data type for variables that are the same as get and set properties but the difference is that a Data Contract in WCF is used to serialize and deserialize the complex data. It defines how data types are serialized and deserialized. Using serialization, you can convert an object into a sequence of bytes that can be transmitted over a network. Using de-serialization, you reassemble an object from a sequence of bytes that you receive from a calling application.   
**Example:  
DataContract**

1. **public** **class** Student
2. {
3. **private** **string** \_Name;
4. **private** **string** \_City;
5. [DataMember]
6. **public** **string** Name
7. {
8. **get**
9. {
10. **return** \_Name;
11. }
12. **set**
13. {
14. \_Name = value;
15. }
16. }
17. }

**Message Contract**When an operation contract required to pass a message as a parameter or return value as a message, the type of this message will be defined as message contract. A message contract defines the elements of the message (like as Message Header, Message Body), as well as the message-related settings, such as the level of message security.  
Message contracts give you complete control over the content of the SOAP header, as well as the structure of the SOAP body.  
**Fault Contract:**A fault contract defines errors raised by the service, and how the service handles and propagates errors to its clients. An operation contract can have zero or more fault contracts associated with it.  
The following is the syntax to raise the custom error in WCF:

1. [ServiceContract]
2. **public** **interface** IGetDetailsService
3. {
4. [OperationContract]
5. [FaultContract(**typeof** (Student))]
6. Student GetDetails(**string** Name);
7. }
8. [DataContract]
9. **public** **class** Student
10. {
11. **private** **string** \_Name;
12. **private** **string** \_City;
13. [DataMember]
14. **public** **string** Name
15. {
16. **get**
17. {
18. **return** \_Name;
19. }
20. **set**
21. {
22. \_Name = value;
23. }
24. }
25. [DataMember]
26. **public** **string** City
27. {
28. **get**
29. {
30. **return** \_City;
31. }
32. **set**
33. {
34. \_City = value;
35. }
36. }
37. }

**Question 9: What is Data Contract in WCF?  
Answer:** A data contract is a formal agreement between a service and a client that abstractly describes the data to be exchanged.  
Data contract can be explicit or implicit. Simple type such as int, string etc has an implicit data contract. User defined object are explicit or Complex type, for which you have to define a Data contract using [DataContract] and [DataMember] attribute.  
A data contract can be defined as follows:

* It describes the external format of data passed to and from service operations.
* It defines the structure and types of data exchanged in service messages.
* It maps a CLR type to an XML Schema.
* It defines how data types are serialized and deserialized. Through serialization, you convert an object into a sequence of bytes that can be transmitted over a network. Through deserialization, you reassemble an object from a sequence of bytes that you receive from a calling application.
* It is a versioning system that allows you to manage changes to structured data.

1. [DataContract]
2. public class CuboidInfo
3. {
4. }

**Question 10: What is Message Contract in WCF?  
Answer**A Message Contract is used to control the structure of a message body and serialization process. It is also used to send / access information in SOAP headers. By default WCF takes care of creating SOAP messages according to service DataContracts and OperationContracts.   
  
A MessageContract controls the structure of a message body and serialization process. A MessageContract can be typed or untyped and are useful in interoperability cases and when there is an existing message format we must comply with.   
The SOAP header is implemented in the namespace system.web.services.protocol.

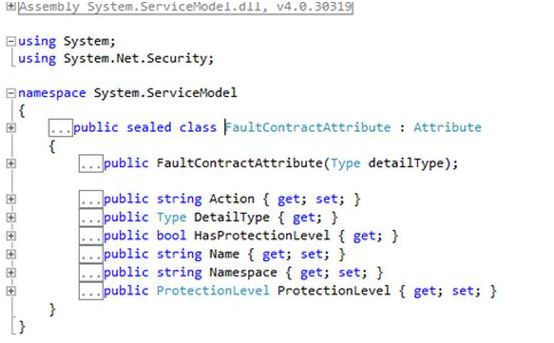
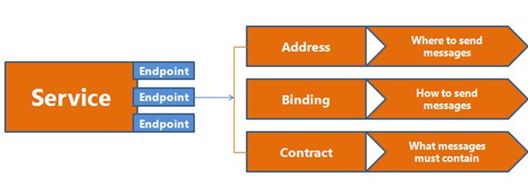
1. [MessageContract]
2. **public** **class** AutherRequest
3. {
4. **public** **string** AutherId;
5. }

**Message Header**  
A Message Header is applied to a member of a message contract to declare the member within the message header.

1. [MessageContract]
2. **public** **class** AutherRequest
3. {
4. [MessageHeader]
5. **public** **string** AutherId;
6. }

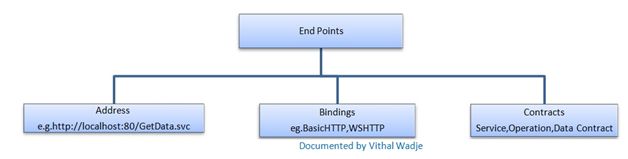
**Message Body Member**  
A Message Body Member is applied to the member of a message contract to declare the members within the message body.

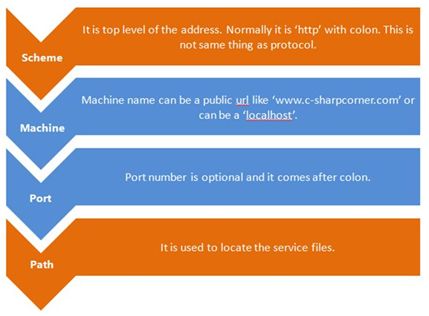
1. [MessageContract]
2. **public** **class** AuthorResponse
3. {
4. [MessageBodyMember]
5. **public** Auther Obj;
6. }

**Question 11: What is Fault Contract in WCF?  
Answer**Fault Contract provides documented view for error accorded in the service to client. This help as to easy identity the what error has occurred, and where. By default when we throw any exception from service, it will not reach the client side. The less the client knows about what happened on the server side, the more dissociated the interaction will be, this phenomenon (not allowing the actual cause of error to reach client) is known as error masking. By default all exceptions thrown on the service side always reach the client as FaultException, as by having all service exceptions indistinguishable from one another, WCF decouples the client from service.   
**Syntax**  
*[OperationContract]  
[FaultContract(typeof(MyException))]  
string getDetails(int value);*  
  
**Question 12: What is End Points and how many types of End points?  
Answer**Endpoints provide the configuration required for the communication and create the complete WCF service application.  
An Endpoint is a piece of information that tells WCF how to build the runtime communication channels to send and receive messages. An endpoint consists of the three things address, binding and contract.   
  
All communication with a Windows Communication Foundation (WCF) service occurs through the endpoints of the service. Endpoints provide clients access to the functionality offered by a WCF service.  
Each endpoint consists of four properties:

* An address that indicates where the endpoint can be found.
* A binding that specifies how a client can communicate with the endpoint.
* A contract that identifies the operations available.

The Structure of an Endpoint-  
Each endpoint consists of the following:

* A set of behaviors that specify local implementation details of the endpoint.
* **Address:** The address uniquely identifies the endpoint and tells potential consumers of the service where it is located. It is represented in the WCF object model by the [EndpointAddress](https://msdn.microsoft.com/en-us/library/system.servicemodel.endpointaddress(v=vs.110).aspx) class. An EndpointAddress class contains:
  + A Uri property, which represents the address of the service.
  + An Identity property, which represents the security identity of the service and a collection of optional message headers. The optional message headers are used to provide additional and more detailed addressing information to identify or interact with the endpoint.
* **Binding:** The binding specifies how to communicate with the endpoint. This includes:
  + The transport protocol to use (for example, TCP or HTTP).
  + The encoding to use for the messages (for example, text or binary).
  + The necessary security requirements (for example, SSL or SOAP message security).
* **Contracts:** The contract outlines what functionality the endpoint exposes to the client. A contract specifies:
  + What operations can be called by a client.
  + The form of the message.
  + The type of input parameters or data required to call the operation.
  + What type of processing or response message the client can expect.  
      
    

**13: What is Address in WCF?  
Answer:** Address - Where - Where to send messages  
An Address is a unique Uniform Resource Locator (URI) that identifies the location of the service. It defines the network address for sending and receiving the messages. It is divided into four parts:  
  
  
  
In WCF every service are associated with unique address. The address provided two imported elements the location of services and the transport protocol which will help to communicate with the services .WCF support the following transport schemes. Here is the answer for what are the supported transport protocols in WCF.

* HTTP/HTTPS
* TCP
* IPC
* Peer Network
* MSMQ
* Service BUS

Let us see here **what is the format of an address in WCF?**  
Address always have the following format.  
*[transport]://[machine or domain][:optional port].  
ex: https://localhost:82/MyService*  
**14: What is Binding in WCF?What are the types of binding?  
Answer:** Binding describes how client will communicate with service. There are different protocols available for the WCF to communicate to the Client. You can mention the protocol type based on your requirements.  
A binding has several characteristics, including the following:

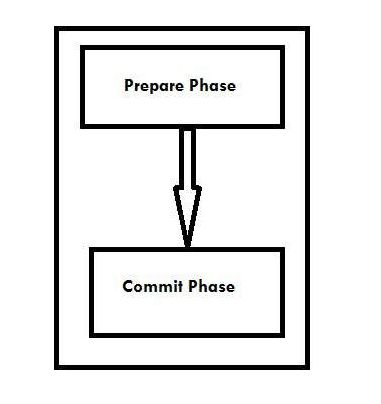
* **Transport -** Defines the base protocol to be used like HTTP, Named Pipes, TCP, and MSMQ are some type of protocols.
* **Encoding (Optional) -** Three types of encoding are available-Text, Binary, or Message Transmission Optimization Mechanism (MTOM). MTOM is an interoperable message format that allows the effective transmission of attachments or large messages (greater than 64K).
* **Protocol(Optional) -** Defines information to be used in the binding such as Security, transaction or reliable messaging capability.

The following table gives some list of protocols supported by WCF binding.

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

**15: Explain transactions in WCF?  
Answer:**   
A transaction is a logical unit of work consisting of multiple activities that must either succeed or fail together. For example, you are trying to make an online purchase for a dress; your amount is debited from the bank, but your ordered dress was not delivered to you. So what did you get from the preceding example? Bank operations hosted in a separate service and the online purchase service (Ordering the goods) hosted as another separate service. For the preceding example the online purchase service did not work well as expected due to the database query error, without knowing it the Bank operations were completed perfectly. For this we really need transactions to ensure that either of the operations should succeed or fail. Before we go to the WCF transactions, let us discuss what exactly are the basics behind the transactions.  
The following are the types of the Transactions:

1. Atomic
2. Long Running

**Phases of WCF Transaction:**There are two phases in a WCF transaction as shown in the following figure.  


* **Prepare Phase -** In this phase, the transaction manager checks whether all the entities are ready to commit for the transaction or not.
* **Commit Phase -** In this phase, the commitment of entities get started in reality.

**Question 16: What are the possible ways of hosting a WCF service?  
Answer:** A WCF service can be self-hosting using a console application or Windows Forms applications. Hosting a WCF service in any managed .Net application is called self-hosting. Now the following is the procedure for creating a WCF service and hosting it in a console application. Hosting Environment Requirements:

* **Availability:** When do you want to be able to reach your service?
* **Reliability:** What happens when your service somehow breaks? How does this affect other consumers?
* **Manageability:** Do you need easy access to information about what is happening on the host where WCF services live?
* **Versioning:** Do you need to support older versions of the service? Do you know who is consuming your services?
* **Deployment:** What is your deployment model? Are you installing through the Microsoft Installer process and Visual Studio deployment packages, or is xcopy sufficient?
* **State:** Are your services stateless? Do you need sessions?

The following are the various options available for hosting a WCF Service:

1. Self-Hosting
2. Windows Service
3. Internet Information Services (IIS)
4. Windows Activation Services (WAS)

**Question 17: What is Self Hosting in WCF?  
Answer:**   
We have our project (it may be Windows or a web application or something else) and we want to host one WCF service within this solution locally. This type of hosting is called Self-Hosting. To implement self-hosting we need to include System.Service.Model.ServiceHost namespace.  
The following are the advantages of self-hosting:

* **Is easy to use:** With only a few lines of code you have your service running.
* **Is flexible:** You can easily control the lifetime of your services through the **Open()** and **Close()** methods of **ServiceHost<T>**.
* **Is easy to debug:** Debugging WCF services that are hosted in a self-hosted environment provides a familiar way of debugging, without having to attach to separate applications that activate your service.
* **Is easy to deploy:** In general, deploying simple Windows applications is as easy as xcopy. You don't need any complex deployment scenarios on server farms, and the like, to deploy a simple Windows application that serves as a WCF **ServiceHost**.
* **Supports all bindings and transports:** Self-hosting doesn't limit you to out-of-the-box bindings and transports whatsoever. On Windows XP and Windows Server 2003, IIS limits you to HTTP only.

The following are the disadvantages of self-hosting:

* **Limited availability:** The service is reachable only when the application is running.
* **Limited features:** Self-hosted applications have limited support for high availability, easy manageability, robustness, recoverability, versioning, and deployment scenarios. At least, out-of-the-box WCF doesn't provide these, so in a self-hosted scenario you have to implement these features yourself; IIS, for example, comes with several of these features by default.

**Question 18: What are the main components of WCF Service?  
Answer:** **What a Service is,**  
A Service is a program or application that is used by other applications. In other words, a service that is directly consumed by the end user to do their work and is something they ask for and recognize Or It's just self-contained business functionality.  
**What a WCF Service is:**Windows Communication Foundation (WCF) was introduced in .NET 3.0. This is a great network distributed system developed by Microsoft for communication between applications. WCF is meant for designing and deploying distributed applications under a Service-Oriented Architecture (SOA) implementation. From MSDN: Windows Communication Foundation is a part of the .NET Framework that provides a unified programming model for rapidly building service-oriented applications that communicate across the web and the enterprise. WCF is a combined feature of Web Service, Remoting, MSMQ and COM+.   
There are the following three main components of a WCF application.

1. **WCF Service class:** A WCF Service class implements some service as a set of methods.
2. **WCF Service host:** WCF supports four types of hosting, IIS, Windows Process Activation Services (WAS), self-hosting and Windows Services.
3. **WCF Service endpoints:** All communication with a Windows Communication Foundation (WCF) service occurs through the endpoints of the service. Usually the name of the Interface will be specified in the contract, so the client application will be aware of the operations that are exposed to the client.

**Each endpoint contains:**

* **Address:** An address that indicates where to find the service.
* **Binding:** A binding that specifies how a client can communicate with the service.
* **Contract:** A Contract that specifies what can the service do for us.

**Question 19: How do we host a WCF service in IIS?  
Answer 8:** IIS hosting: if we are hosting a WCF Service in IIS (version 5 or 6), only the HTTP option is available as the transport protocol. In later versions of Internet Information Services (IIS), we can use any other protocol (TCP/IP, MSMQ, NamedPipes and so on.) as transport protocol.  
If we host our service under IIS then we have all the features of IIS like process recycling, ideal shutdown etc. Visual Studio provides two different ways to host our WCF service in a local IIS in all are familiar with Publishing Wizard. Here we will discuss the second way using Visual Studio only in a simple manner to host our WCF service in local IIS. To host the WCF services follow the steps below.  
The updated configuration for System.ServiceModel in the web.config will be as follows:

1. <system.serviceModel>
2. <behaviors>
3. <serviceBehaviors>
4. <behavior name=”StudentServiceBehavior”>
5. <serviceMetadata httpGetEnabled=”**true**”/>
6. <serviceDebug includeExceptionDetailInFaults=”**false**”/> </behavior>
7. </serviceBehaviors>
8. </behaviors>
9. <services>
10. <service behaviorConfiguration=”StudentServiceBehavior” name=”StudentService.StudentService”>
11. <endpoint address=”http://localhost/StudentIISHost/MyStudentHost.svc” binding=”wsHttpBinding” contract=”StudentService.IStudentService”>
12. <identity>
13. <dns value=”localhost”/> </identity>
14. </endpoint>
15. <endpoint address=”mex” binding=”mexHttpBinding” contract=”IMetadataExchange”/> </service>
16. </services>
17. </system.serviceModel>

**Question 20: What is one-way operation?  
Answer:** When an operation has no return value, and the client does not care about the success or failure of the invocation. WCF offers one-way operations to support this sort of fire-and-forget invocation,: once the client issues the call, WCF generates a request message, but no correlated reply message will ever return to the client. The one-way message exchange pattern is useful when a client needs to send information to a service but doesn't receive a response.  
Use the one-way design pattern:

* When the client must call operations and is not affected by the result of the operation at the operation level.
* When using the NetMsmqBinding or the MsmqIntegrationBinding class. (For more information about this scenario, see Queues in Windows Communication Foundation.)  
    
  To create a one-way service contract, define your service contract, apply the OperationContractAttribute class to each operation, and set the [IsOneWay](https://msdn.microsoft.com/en-us/library/system.servicemodel.operationcontractattribute.isoneway(v=vs.110).aspx) property to true, as shown in the following sample code.

1. [ServiceContract(Namespace = "http://Microsoft.ServiceModel.Samples")]
2. **public** **interface** IOneWayCalculator
3. {
4. [OperationContract(IsOneWay = **true**)]
5. **void** Add(**double** n1, **double** n2);
6. [OperationContract(IsOneWay = **true**)]
7. **void** Subtract(**double** n1, **double** n2);
8. [OperationContract(IsOneWay = **true**)]
9. **void** Multiply(**double** n1, **double** n2);
10. [OperationContract(IsOneWay = **true**)]
11. **void** Divide(**double** n1, **double** n2);
12. }

**Question 21: Why we need One Way Service?**  
**Aswer:** There might be requirements where Service's Operation is not going to return any value. And at the same time client also does not bother about success or failure of service. Only for the client does calling the service matter. Client will call the operation and forget and continue with the operations.  
**One Way Operations:**This is the way in WCF to achieve Fire and forget invocation of the service. Once the Client issues the call, WCF generates the request message and there is no corresponding response message for that. No Service side Exception will make their way to client.   
**Configuring One Way Operation**  
Boolean **ISOneWay** property of **OperationContract** class is used to configure service as one way service. We need to set this property to true. Setting true to this property turns a service operation as one way operation.

1. **using** System;
2. **using** System.Net.Security;
3. **namespace** System.ServiceModel
4. {
5. [AttributeUsage(AttributeTargets.Method)]
6. **public** **sealed** **class** OperationContractAttribute: Attribute
7. {
8. **public** OperationContractAttribute();
9. **public** **string** Action
10. {
11. **get**;
12. **set**;
13. }
14. **public** **bool** AsyncPattern
15. {
16. **get**;
17. **set**;
18. }
19. **public** **bool** HasProtectionLevel
20. {
21. **get**;
22. }
23. **public** **bool** IsInitiating
24. {
25. **get**;
26. **set**;
27. }
28. **public** **bool** IsOneWay
29. {
30. **get**;
31. **set**;
32. }
33. **public** **bool** IsTerminating
34. {
35. **get**;
36. **set**;
37. }
38. **public** **string** Name
39. {
40. **get**;
41. **set**;
42. }
43. **public** ProtectionLevel ProtectionLevel
44. {
45. **get**;
46. **set**;
47. }
48. **public** **string** ReplyAction
49. {
50. **get**;
51. **set**;
52. }
53. }
54. }

The client has nothing to do anything specific to invoke one way operation.  
  
**Question 22: Explain what is DataContractSerializer?  
Answer:** WCF provides a message-oriented programming framework. We use WCF to transmit messages between applications. Internally WCF represents all the messages by a Message class. When WCF transmits a message it takes a logical message object and encodes it into a sequence of bytes. After that WCF reads those bytes and decodes them in a logical object. The process forms a sequence of bytes into a logical object; this is called an encoding process. At runtime when WCF receives the logical message, it transforms them back into corresponding .Net objects. This process is called serialization.   
  
  
  
WCF supports three basic serializers:

* XMLSerializer
* NetDataContractSerializer
* DataContractSerializer

WCF deserializes WCF messages into .Net objects and serializes .Net objects into WCF messages. WCF provides DataContractSerializer by default with a servicecontract. We can change this default serializer to a custom serializer like XMLSerializer.

1. [XmlSerializerFormat]
2. [ServiceContract]
3. **public** **interface** IService1
4. {
5. [OperationContract]
6. **void** AddAuthor(Author author);
7. }

The XmlSerializerFormat attribute above the ServiceContract means this serializer is for all operation contracts. You can also set a separate contract for each operation.  
 **XMLSerializer**  
  
We can find XMLSerializer in the System.Xml.Serialization namespace. WCF supports this serialization from .Net 1.0. It is used by default with the ASP.Net webservices (ASMX).  
  
**NetDataContractSerializer**  
  
NetDataContractSerializer is analogous to .Net Remoting Formatters. It implements IFormatter and it is compatible with [Serializable] types. It is not recommended for service oriented design.  
  
**DataContractSerializer**  
  
DataContractSerializer is a default serializer for WCF. We need not to mention DataContractSerializer attribute above the service contract.   
  
**Question 23: What is Transaction Propagation? And how WCF support it?  
Answer:** Transaction propagation is the ability to propagate a 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, WCF supports SOA, so it provides support for transaction propagation as well.  
To enable transaction propagation, we need to set the value of the TransactionFlow property of the binding being used. This can be done programmatically as follows:

1. WSHttpBinding bindingBeingUsed = **new** WSHttpBinding();
2. bindingBeingUsed.TransactionFlow = "true";

Or it can be done decoratively by updating the configuration file as follows:

1. <bindings>
2. <wsHttpBinding>
3. <binding name="binding1" transactionFlow="true" />
4. </wsHttpBinding>
5. </bindings>

The default value for the TransactionFlow property is "**False**".  
  
**Do all WCF bindings support Transaction Propagation?**  
No. Not all WCF bindings support transaction propagation. Only the following list of bindings support it:

* wsHttpBinding
* netTcpBinding
* netNamedPipeBinding
* wsDualHttpBinding
* wsFederationHttpBinding

**Question 24: What is WCF throttling?  
Answer 14:** WCF throttling provides some properties that you can use to limit how many instances or sessions are created at the application level. Performance of the WCF service can be improved by creating proper instance.  
The throttling of services is another key element for WCF performance tuning. WCF throttling provides the prosperities maxConcurrentCalls, maxConcurrentInstances, and maxConcurrentSessions, that can help us to limit the number of instances or sessions are created at the application level.

|  |  |
| --- | --- |
| **Attribute / Property** | **Description** |
| maxConcurrentCalls | This specifies the maximum number of messages processed across the service host. The default value for this property is 16 (WCF 4.0 is improved to default is 16 \* Processor Count). |
| maxConcurrentInstances | This specifies the maximum number of instances of a context object that executes at one time with the service. The default is Int32.MaxValue. |
| maxConcurrentSessions | This specifies the maximum number of sessions at one time within the service host object. The default value is 10 (WCF 4.0 increases that to 100 \* Processor Count). |

1. <configuration>
2. <system.serviceModel>
3. <services>
4. <service .... .... .... </service>
5. </services>
6. <behaviors>
7. <serviceBehaviors>
8. <behavior name="ServiceBehavior">
9. <serviceThrottling maxConcurrentCalls="16" maxConcurrentSessions="100" maxConcurrentInstances="10" />
10. <serviceMetadata httpGetEnabled="true" /> </behavior>
11. </serviceBehaviors>
12. </behaviors>
13. </system.serviceModel>
14. </configuration>

**Question 25: What is a Service Proxy in WCF?  
Answer:** A service proxy or simply proxy in WCF enables application(s) to interact with a 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 a service contract (signature only, not the implementation).  
A client application uses the **WCF** client **proxy** to communicate with the service. Client applications usually import a **service's** metadata to generate **WCF** client code that can be used to invoke the **service**.   
The basic steps for creating a **WCF** client include the following: Compile the service code. The proxy provides the same operation as service contract, but also provides additional methods for managing proxy life cycle and the connection to the service. Proxy contains all the information of the service like Address, Binding and Configuration so that client knows how to communicate with the service. What security policies he has to follow and all whatever is dictated by the service to work properly.

**Question 26: What is WCF Concurrency and How many modes are of Concurrency in WCF?  
Answer:** WCF Concurrency means “Several computations are executing simultaneously.  
WCF concurrency helps us configure how WCF service instances can serve multiple requests at the same time. You will need WCF concurrency for the below prime reasons; there can be other reasons as well but these stand out as important reasons:  
We can use the concurrency feature in the following three ways:

* Single
* Multiple
* Reentrant

**Single:** A single request will be processed by a single thread on a server at any point of time. The client proxy sends the request to the server, it process the request and takes another request.  
The following is the declaration of Concurrency.Single:  
Concurrency  
**Multiple:** Multiple requests will be processed by multiple threads on the server at any point of time. The client proxy sends multiple requests to the server. Requests are processed by the server by spawning multiple threads on the server object.   
**Reentrant:** The reentrant concurrency mode is nearly like the single concurrency mode. It is a single-threaded service instance that receives requests from the client proxy and it unlocks the thread only after the reentrant service object calls the other service or can also call a WCF client through call back.

1. [ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Multiple]
2. Public **class** MyService : IMyService
3. {
4. }

For more details visit the following link:

* [Instancing and Concurrency in WCF](http://www.c-sharpcorner.com/UploadFile/rkartikcsharp/instancing-and-concurrency-in-wcf/)

**Question 27: What is REST and how to create a WCF RESTful Service ?  
Answer: REST-**Representational State Transfer (REST) is a protocol for exchanging data over a distributed environment. The main idea behind REST is that we should treat our distributed services as a resource and we should be able to use simple HTTP protocols to perform various operations on that resource.  
  
**JSON**  
The **JavaScript Object Notation (JSON)** data format, or JSON for short, is derived from the literals of the JavaScript programming language. **JSON** helps us to present and exchange data in a self-descriptive, independent and light way. This data can then be easily consumed and transformed into JavaScript objects.  
  
In most browser-based applications, WCF can be consumed using JavaScript, jQuery, AngularJS and so on. When a client makes a call to WCF, JSON or XML is used as the format of the communication. WCF has the option to send the response in JSON object. This can be configured with the WebGet orWebInvoke attribute and the WebHttpBinding. This allows you to expose a ServiceContract as a REST service that can be invoked using either JSON or plain XML.

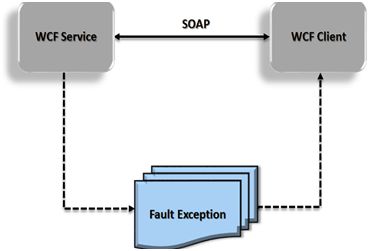
* GET: Retrieves the information.
* PUT: Replaces the entire collection with another collection.
* POST: Creates a new entry in the collection.
* DELETE: Deletes the entire collection.

For more details visit the following link:

* [Create a WCF RESTful Service](http://www.c-sharpcorner.com/UploadFile/47548d/create-a-wcf-restful-service/)

**Question 28: What is Exception Handling in WCF? What are the ways for WCF Exception Handling?**  
**Answer:** Exception handling is critical to all applications for troubleshooting the unexpected problems in applications. The Windows Communication Framework (WCF) provides several options for handling exceptions in WCF services. This article discusses these approaches and describes the advantages and disadvantages of each. The following options are provided to handle exceptions in WCF:

1. Using returnUnknownExceptionsAsFaults: **Debugging Mode**
2. Using FaultException: **Best Option.**
3. Using IErrorHandler: **Only when the exception can't be handled by Fault**

**Exception handling in WCF**  
  
  
  
**Exceptions inside a WCF Service**  
Before describing the details of exception handling in WCF, let's explore what happens if we do not handle an exception inside the service. Consider a service with the CreateUser method as shown in the following:

1. public **void** CreateUser(User user)
2. {
3. if(user.isValid())
4. {
5. //Create User
6. }
7. else
8. {
9. throw **new** ApplicationException(“User Inavalid.”);
10. }
11. }

**Example**

1. **public** **class** DBManagerService: IDBManagerService
3. **void** Save(Employee emp)
4. {
5. try
6. {
7. //Code to store an employee object to the database
8. }
9. **catch** (Exception ex
10. {
11. **throw** **new** Exception(“Error occurred
12. **while** saving data…”);
13. }
14. }
15. }

For more details visit the following link:

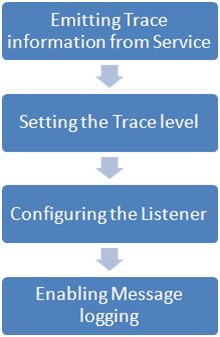
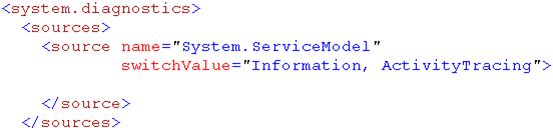
* [Best Ways For WCF Exception Handling](http://www.c-sharpcorner.com/UploadFile/aravindbenator/wcf-exception-handling-best-ways/)

**Question 29: What is Tracing in WCF?**  
**Answer:** The tracing mechanism in the Windows Communication Foundation is based on the classes that reside in the System.Diagnostic namespace. The important classes are Trace, TraceSource and TraceListener.  
  
Configuring WCF to emit tracing information/Define Trace Source, we have the following options:

* System.ServiceModel
* System.ServiceModel.MessageLogging
* System.ServiceModel.IdentityModel
* System.ServiceModel.Activation
* System.Runtime.Serialization
* System.IO.Log
* Cardspace

**Example**

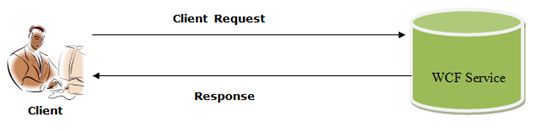
1. configuration>
2. <system.diagnostics>
3. <sources>
4. <source name="System.ServiceModel"
5. switchValue="Information, ActivityTracing"
6. propagateActivity="true">
7. <listeners>
8. <add name="traceListener"
9. type="System.Diagnostics.XmlWriterTraceListener"
10. initializeData= "c:\log\Traces.svclog" />
11. </listeners>
12. </source>
13. </sources>
14. </system.diagnostics>
15. </configuration>

There are essentially four steps involved in WCF Tracing:  
  
  
  
Set the Trace Level:  
  
  
  
For more details visit the following link:

* [Tracing in WCF: Understanding Basic Steps](http://www.c-sharpcorner.com/UploadFile/dhananjaycoder/tracing-in-wcf-understanding-basic-steps/)

**Question 30: What are the Various Types of Bindings WCF Supports?**  
**Answer:** Binding describes how a client is going to communicate to WCF service. Binding is used as per client need. It supports different types of protocol to communicate with client and different types of encoding to transfer the data over the internet. So, basically binding is nothing but it is a way of communication between client and service as per client need.  
  
**Basic binding**  
This binding is provided by the BasicHttpBinding class. It is designed to expose a WCF service as an ASMX web service, so that old clients (that are still using an ASMX web service) can consume the new service. By default, it uses the HTTP protocol for transport and encodes the message in UTF-8 text format. You can also use HTTPS with this binding.  
 **Web binding**  
This binding is provided by the WebHttpBinding class. It is designed to expose WCF services as HTTP requests using HTTP-GET and HTTP-POST. It is used with REST based services that may provide output in XML or JSON format. This is very much used with social networks for implementing a syndication feed.  
 **Web Service (WS) binding:**  
This binding is provided by the WSHttpBinding class. It is like a basic binding and uses HTTP or HTTPS protocols for transport. But this is designed to offer various WS - \* specifications such as WS – Reliable Messaging, WS - Transactions, WS - Security and so on which are not supported by Basic binding.  
  
wsHttpBinding= basicHttpBinding + WS-\* specification  
 **WS Dual binding:**  
This binding is provided by the WsDualHttpBinding class. It is like a WsHttpBinding except it supports bi-directional communication which means both clients and services can send and receive messages.  
 **TCP binding:**  
This binding is provided by the NetTcpBinding class. It uses TCP protocol for communication between two machines within intranet (means same network). It encodes the message in binary format. This is a faster and more reliable binding compared to the HTTP protocol bindings. It is only used when the communication is WCF-to-WCF which means both client and service should have WCF.  
 **IPC binding:**  
This binding is provided by the NetNamedPipeBinding class. It uses named pipe for communication between two services on the same machine. This is the most secure and fastest binding among all the bindings.  
  
**MSMQ binding:**  
This binding is provided by the NetMsmqBinding class. It uses MSMQ for transport and offers support to a disconnected message queued. It provides solutions for disconnected scenarios in which the service processes the message at a different time than the client sending the messages.  
 **Federated WS binding:**  
This binding is provided by the WSFederationHttpBinding class. It is a specialized form of WS binding and provides support to federated security.  
  
For more details visit the following link:

* [Different Types Of Bindings In WCF](http://www.c-sharpcorner.com/UploadFile/8a67c0/different-types-of-binding-in-wcf/)  
    
  OR
* [Various Types of Bindings WCF Supports](http://www.c-sharpcorner.com/UploadFile/87b416/type-of-bindings-that-wcf-supports/)

**Question 31: What is Instance Context Mode in WCF?**  
**Answer:** An Instance Context mode defines how long a service instance remains on the server.  
  
**Request Response Cycle**  
  
  
  
Whenever the client sends a request to a WCF service, what exactly happens behind the scenes? Basically after making a client request the WCF service will create a service class instance at the service that will do the operations involved and then it will return the response back to the client. In this request and response process the service instance object has been created in the process.  
  
The WCF framework has defined the following three Instance Context modes:  
  
**PerCall:** A new instance of the service will be created for the request from the same client or a different client, meaning every request is a new request. In this mode no state is maintained.In per-call service, every client request achieves a new dedicated service instance and its memory consumption is less as compared to other types of instance activation.

1. [ServiceContract]
2. interfaceIMyContract
3. {...}
4. [ServiceBehavior(InstanceContextMode=InstanceContextMode.PerCall)]
5. classMyService:IMyContract
6. {...}

**Per-Session Service:** A new Instance will be created for every new client session and the scope of that object will be the scope of that session.

1. [ServiceBehavior(InstanceContextMode=InstanceContextMode.PerSession)]
2. classMyService:IMyContract
3. {...}

**Singleton Service:** A single instance will be created for the service object that will take care of all the requests coming from the same client or a different one.  
  
By decorating the service with a service behavior, an Instance Context mode can be set.  
  
[ServiceBehavior(InstanceContextMode=InstanceContextMode.PerCall)]   
  
For more details visit the following link:

* [Instance Context Mode in WCF](http://www.c-sharpcorner.com/UploadFile/de41d6/instance-context-mode-in-wcf/)

**Question 32: What is the KnownType Attribute in WCF?  
  
Answer:** The KnownTypeAttribute class allows you to specify, in advance, the types that should be included for consideration during deserialization.  
  
The WCF service generally accepts and returns the base type. If you expect the service to accept and return an inherited type then we use the knowntype attribute.  
  
When passing parameters and return values between a client and a service, both endpoints share all of the data contracts of the data to be transmitted.

1. [DataContract]
2. [KnownType(**typeof** (Student))]
3. [KnownType(**typeof** (Teacher))]
4. publicabstractclass Person
5. {
6. [DataMember]
7. publicint Code
8. {
9. **get**;
10. **set**;
11. }
12. [DataMember]
13. publicstring Name
14. {
15. **get**;
16. **set**;
17. }
18. }

For more details visit the following link:

* [KnownType Attribute in WCF: Part 6](http://www.c-sharpcorner.com/UploadFile/b182bf/knowntype-attribute-in-wcf-part-6/)

**Question 33: How to create Basic HTTP Binding in WCF?**  
**Answer: BasicHttpBinding,**  
The **basicHttpBinding** binding is used in order to provide support for older clients that expect a legacy ASMX Web service. The **TransportCredentialOnly** security mode option passes the user credentials without encrypting or signing the messages.  
  
BasicHttpBinding is suitable for communicating with ASP.NET Web Service (ASMX) based services that conform to the WS-Basic Profile that conforms with Web Services.

* This binding uses HTTP as the transport and text/XML as the default message encoding.
* Security is disabled by default.
* This binding does not support WS-\* functionalities like WS- Addressing, WS-Security, WS-ReliableMessaging.
* It is fairly weak on interoperability.

Use the following procedure.

* Create a WCF basicHttpBinding project.
* Start Visual Studio and select New Project from the Start page or from the File menu, select New and then Project.
* After selecting, a dialog will pop up showing all the templates provided by Visual Studio.
* From the Installed Templates select C# then inside that select WCF and inside that you will select WCF Service Application.

For more details visit the following link:

* [How to Create Basic HTTP Binding in WCF](http://www.c-sharpcorner.com/UploadFile/4d9083/how-to-create-basic-http-binding-in-wcf/)

**Question 34: What is the difference between BasicHttpBinding and WsHttpBinding?**  
**Answer:** Below is a detailed comparison table between both the entities from security, compatibility, reliability, and SOAP version perspectives.

|  |  |  |
| --- | --- | --- |
| **Criteria** | **BasicHttpBinding** | **WsHttpBinding** |
| Security support | This supports the old ASMX style, i.e., WS-BasicProfile 1.1. | This exposes web services using WS-\* specifications. |
| Compatibility | This is aimed for clients who do not have .NET 3.0 installed and it supports wider ranges of clients. Many of the clients like Windows 2000 still do not run .NET 3.0. So an older version of .NET can consume this service. | As it is built using WS-\* specifications, it does not support wider ranges of clients and it cannot be consumed by older .NET versions less than 3 version. |
| SOAP version | SOAP 1.1 | SOAP 1.2 and WS-Addressing specification. |
| Reliable messaging | Not supported. In other words, if a client fires two or three calls you really do not know if they will return back in the same order. | Supported as it supports WS-\* specifications. |
| Default security options | By default, there is no security provided for messages when the client calls happen. In other words, data is sent as plain text. | As WsHttBinding supports WS-\*, it has WS-Security enabled by default. So the data is not sent in plain text. |
| Security options | • None • Windows – default authentication • Basic • Certificate | • None • Transport • Message • Transport with message credentials |

One of the biggest differences you must have noticed is the security aspect. Bydefault, BasicHttpBinding sends data in plain text while **WsHttpBinding** sends it in an encrypted and secured manner. To demonstrate the same, let's make two services, one using **BasicHttpBinding** and the other using WsHttpBinding and then let's see the security aspect in a more detailed manner.  
  
For more details visit the following link:

* [WCF FAQ Part 3 - 10 Security Related FAQ](http://www.c-sharpcorner.com/UploadFile/shivprasadk/wcf-faq-part-3-10-security-related-faq/)

**Question 35: What is Security Implementation in WCF? How many are there?**  
  
**Answer:** As WCF supports various protocols i.e. TCP, HTTP, and MSMQ, user must be sure enough to take necessary steps to guard your message and also must establish security policies for protecting messages and for authenticating and authorizing calls. WCF provide a very easy and rich configurable environment to implement security.WCF supports following securities:

* Message
* Transport
* TransportWithMessageCredential

**Message Security**  
Message security uses the WS-Security specification to secure messages. The message is encrypted using the certificate and can now safely travel over any port using plain http. It provides end-to-end security.  
  
**Transport Security**  
Transport security is a protocol implemented security so it works only point to point. As security is dependent on protocol, it has limited security support and is bounded to the protocol security limitations.  
  
**TransportWithMessageCredential:**  
This we can call a mixture of both Message and Transport security implementation.  
  
For more details visit the following link:

* [WCF Security Implementation](http://www.c-sharpcorner.com/UploadFile/sam_manila/wcf-security-implementation/)  
    
  OR
* [Basics Of WCF Security](http://www.c-sharpcorner.com/UploadFile/rkartikcsharp/basics-on-wcf-security/)

**Question 36: What is Streaming in WCF?**  
**Answer:** In WCF any receiving message is delivered only once entire message has been received. What I mean here is that first message is buffered at the receiving side and once it is fully received it gets delivered to the receiving end. Main problem with this approach is that receiver end is unresponsive while message is getting buffered. So default way of message handling in WCF is ok for small size messages but for the large size messages this approach is not good. So to overcome this problem Streaming in WCF come into action.  
  
**Steaming and Binding**

1. TCP, IPC and HTTP bindings support streaming.
2. For all the Binding streaming is disabled by default.
3. Streaming of the message should be enabled in the binding to override the buffering and enable the streaming.
4. TransferMode property should be set according to the desired streaming mode in the bindings.
5. Type of TransferMode property is enum TransferMode

**Configuring Streaming in Config file**

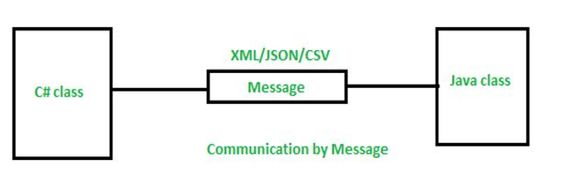
1. <system.serviceModel>
2. <services>
3. <service name="OneWayService.Service1" behaviorConfiguration="OneWayService.Service1Behavior">
4. <!-- Service Endpoints -->
5. <endpoint address="" binding="basicHttpBinding" bindingConfiguration="StreamedHttp" contract="OneWayService.IService1">
6. <identity>
7. <dns value="localhost" /> </identity>
8. </endpoint>
9. <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" /> </service>
10. </services>
11. <bindings>
12. <basicHttpBinding>
13. <binding name="StreamedHttp" transferMode="Streamed" /> </basicHttpBinding>
14. </bindings>
15. <behaviors>
16. <serviceBehaviors>
17. <behavior name="OneWayService.Service1Behavior">
18. <serviceMetadatahttpGetEnabled="true" />
19. <serviceDebugincludeExceptionDetailInFaults="false" /> </behavior>
20. </serviceBehaviors>
21. </behaviors>
22. </system.serviceModel>

For more details visit the following link:

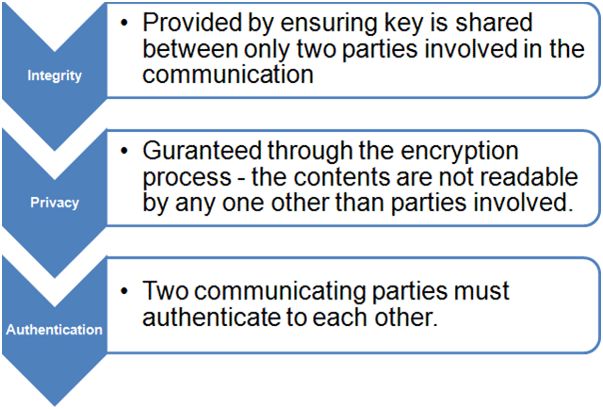
* [Streaming in WCF](http://www.c-sharpcorner.com/UploadFile/dhananjaycoder/streaming-in-wcf/)

**Question 37: What is SOA, and Messages in WCF?**  
  
Answer: **SOA is nothing but an architectural style where two non-compatible applications can** communicate using a common language and WCF is nothing but one example of the Service Oriented Architecture (SOA).Characteristics of SOA:

* In SOA, Services should be **independent** of other services. Altering a service should not affect calling service.
* Services should be able to **define themselves**. Services should be able to answer a question what is does? It should be able to tell client what all operations it does, what all data types it uses and what kind of responses it will return.
* Services should support **reliable messaging**. Means there should be a guarantee that request will be reached to correct destination and correct response will be obtained.
* Services should support **secure communication.**

**Message Pattern:** A message in WCF is very similar to messages in the real world.  
  
It describes how the programs will exchange message each other. There are three way of communication between source and destination,  
  
There is no need to call a Java class from a C# class. When they want to talk with each other they just send a message to each other. In the Next programming industry though it's fine and cool to communicate via messages but there should be some standardization of those messages. Then, when people try to exchange data between two programs they send the standardized message to get it done.  
  
  
  
For more details visit the following link:

* [Understand WCF: Part-1: What is SOA, Services and Messages](http://www.c-sharpcorner.com/UploadFile/dacca2/understand-wcf-part-1-what-is-soa-services-and-messages/)

**Question 38: What is Transport level security in WCF?**  
  
**Answer:** The goal of transport level security is to provide integrity, privacy and authentication.  
  
  
  
Advantage of using Transport security:

* Less chances of sniffing network.
* Less chances of Phishing network.
* Less chances of message alteration.
* Less chances of replay of message attack.

Regardless of the Binding used, Transport level security provides:

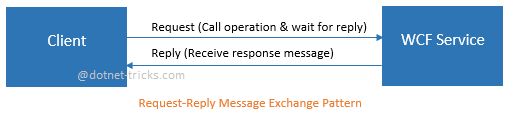
* Authentication of the sender.
* Authentication of the service.
* Message integrity
* Message confidentiality.
* Replay of message detection.

For more details visit the following link:

* [Transport level Security in WCF](http://www.c-sharpcorner.com/uploadfile/dhananjaycoder/transport-level-security-in-wcf/)

**Question 39: What Message Exchange Patterns (MEPs) supported by WCF? Explain each of them briefly.**  
  
**Answer:** MEP stands for Message Exchange Pattern. In WCF we are creating services. What does a service do for us? A service does some task on behalf of us or sends a response back from our request. All such communications are done using messages. We send some message as a request and get a message as a response from the service. WCF supports three types of MEPs:

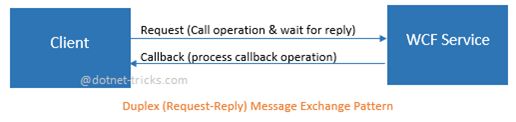
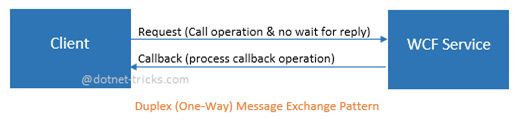
* Request / Response
* One-Way
* Duplex

Message Exchange Patterns describes the way of communication between Client and Server means how client and server would be exchange messages to each other. There are three types of message exchange patterns.  
  
**Request/Response**  
  
Request / Response is a very common MEP. To set this pattern we need to set the IsOneWay property of OperationContractAttribute as false. And the default value of the IsOneWay property is false. So ultimately we do not need to set this property. It is set by default. That is why it is very easy to implement.  
  


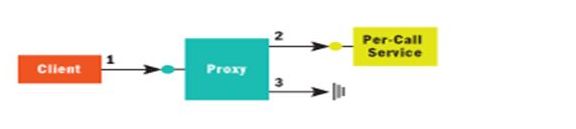
1. [ServiceContract]
2. publicvinterface RequestReplyService
3. {
4. [OperationContract]
5. **string** GetData(**int** value);
7. [OperationContract(IsOneWay = **false**)]
8. voidSaveData(**string** value);
9. }

**One-Way:**  
Set the IsOneWay property of OperationContractAttribute to true for a OneWay message exchange pattern. Suppose you send a message to a service. This pattern is used when the service does some operation and you do not want a response back. For example you want to change the status of a transaction from pending to completed and you do not want to get a confirmation from the service that the status is changed. You can use a OneWay pattern.

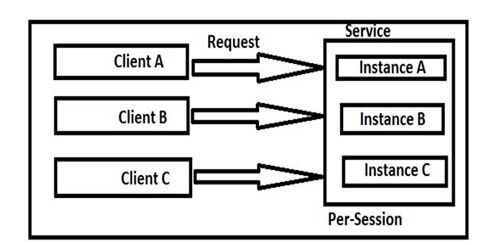
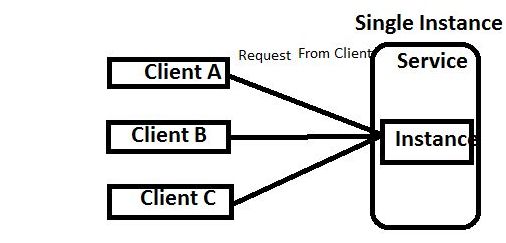
1. [ServiceContract]
2. **public** **interface** IService1
3. {
4. [OperationContract(IsOneWay=**true**)]
5. **void** ChangeTransactionStatus(**int** value);
6. }

  
**Duplex:**The duplex MEP is a two-way message channel. When the client sends a message to the service to instantiate some long-running processing and requires notification back from the service, a duplex MEP is applicable.  
There are two types of contacts, ServiceContract and CallbackContract required to implement a duplex MEP. This MEP is declared by associating a CallbackContract with the ServiceContract. It is done through a CallbackContract property of ServiceContract.  
  
  
  
  
  
**Question 40: Explain briefly different Instance Modes in WCF?  
Answer:** Instance mode is basically a service side implementation detail. It should not manifest itself on the client side. WCF defines the behaviors. Behavior is a local attribute of a service or client that does not affect communication. The client never knows about the service behavior and does not manifest service binding or published metadata.  
Instance Modes available in WCF are:

1. PerCall
2. PerSession
3. Single

PerCall Services  
  
  
When a service is configured PerCall, every client gets a new service instance. The above diagram shows how a PerCall instance works:

1. Client calls the proxy and proxy forwards the call to service.
2. WCF creates an instance and calls the method call.
3. When method call returns, WCF calls IDisponsible if implemented.

**PerSession Service**  
WCF maintains a logical session between client and service in Persession instance mode. A new instance of the service class is created for each client.   
The following given diagram represents the request from the client using Persession service instance mode:  
  
**Single Instance Service**  
When we configure a service as a singleton, all clients are connected to a single instance context. We configure a singleton service by setting the InstanceContextMode property as single. Only one instance is created upon creation of a service host. This instance is forever for the service. Another service instance is created only when we reset the IIS or when the service host is shut down.  
**Diagrammatic representation of a singletion instance:**  
  
  
When should you use per call, per session, and single mode?  
**Per call:**

* You want a stateless service.
* Your service holds intensive resources like connection objects and huge memory objects.
* Scalability is a prime requirement. You would like to have a scaled out architecture.
* Your WCF functions are called in a single threaded model.

**Per session:**

* You want to maintain states between WCF calls.
* You a scaled up architecture.
* Light resource references.

**Single:**

* You want share global data through your WCF service.
* Scalability is not a concern.

For more details visit the following link:

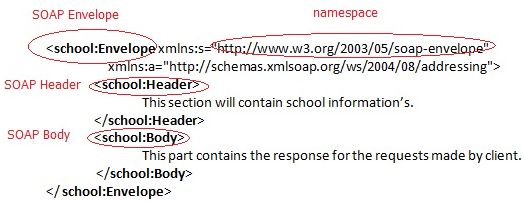
* [Instance Modes in WCF](http://www.c-sharpcorner.com/UploadFile/875c4c/instance-modes-in-wcf/)

**Question 41: Explain bindings in WCF with description?  
Answer:** It specifies how to access the hosted service. There are following characteristics of binding:  
**Transport** defines the communication protocol to be used to communicate between service and client. It may be HTTP, TCP, MSMQ, NamedPipes etc. It is mandatory to define transport.  
**Encoding** defines the technique used to encode the data before communicating it from one end to the other.  
**Protocol** defines the configurations like reliability, security, transaction, timeouts, message size, etc.  
The following table shows the types of bindings and their uses:

|  |  |
| --- | --- |
| **Binding Name** | **Description** |
| BasicHttpBinding | Allows us to create and consume ASMX-style services within WCF. No Security provided for the messages. No Reilable Messaging, Responses will not come in order. Suitable for clients who do not have .Net 3.0 installed. Transport: HTTP. |
| webHTTPBinding | Allows us to create and expose your services as HTTP requests. Used for REST Based Services. |
| wsHttpBinding | Exposes web services using ws-\* specifications. Security provided (Ws-Security) for messages by default. Uses for WS-Transactions, WS-BusinessActivity. |
| wsDualHttpBinding | This is like the preceding one, but it uses duplex contracts. This means the both client and server can both send and receive messages |
| wsFederationHttpBinding | Provides a mechanism for exposing a federated service. The client uses a security token issued by a security token service to authenticate. |
| netTCPBinding | Service to Service Communication in an intranet. Features include transactions and security. For WCF to WCF service communication netTCPBinding is best. |
| netPeerTCPBinding | Used when you require more security for Peer to Peer Communication as netTCPBinding. |
| netMsMqBinding | Binding for asynchronous communication using Microsoft Message Queue (MSMQ). Best when you need to execute service operations in a queued manner. |

**Question 42: What is MSMQ in WCF?  
Answer:** Microsoft Messaging Queue (MSMQ) technology is used for asynchronous communication using messages. MSMQ also can be considered to be an Inter- process communication capability.  
Whenever two processes want to communicate with each other in a "Fire and Forget" manner, MSMQ is very useful for that.  
**Example:** For example what if Billing software needs to process 1000 bills at midnight and must send mail to all those users. Such as when the operator runs the software he wants immediate notification. The operator can't wait for all the bills to be processed and gets email.  
Here MSMQ plays an important role for communication by billing software to send those 1000's of customer email information into the Queue and the Queue event handler will handle the requests. In this way the Operator gets instant notification of the processes instead of knowing the "Behind the Scene" process.  
**Question 43: What is Method overloading in WCF?  
Answer:** Method Overloading is a feature that allows creation of multiple methods with the same name but each method should differ from another in the context of input and output of the function. No two methods must have the same type parameters. Either they must differ in the number of parameters or in the type of the parameters.You can also refer to Method Overloading as a compile-time polymorphism since the calling method knows the address of the called method and the method addresses are resolved at compile time. This is also called as Early Binding.  
**Example:**

1. [ServiceContract]
2. **public** **interface** ICalenderService
3. {
4. [OperationContract(Name = ”GetScheduledEventsByDate”)]
5. ScheduledEvent[] GetScheduledEvents(DateTime date);
6. [OpeartionContract(Name = ”GetScheduledEventsByDateRange”)]
7. ScheduledEvent[] GetScheduledEvents(DateTime start, DateTime end);
8. }

**Question 44: What is WCF Messaging Layer?  
Answer:** Messaging layer is composed of channels. A channel is a component that processes a message in some way. Channels are the core abstraction for sending messages to and receiving messages from an Endpoint.   
**Message Structure:**WCF uses messages to pass data or exchange information from one point to another. All messages are SOAP messages. The basic structures of a SOAP message are made up of three components.  
**SOAP Envelope:**The SOAP envelope is a container for the two most important pieces of a SOAP message, the SOAP header and the SOAP body. A SOAP envelope contains several pieces of key information in the form of elements.   
**SOAP Header:**Using a SOAP header, we can pass useful information about the services to the outer world if needed; it's just for information sharing. Any child elements of the header element are called "header blocks". This provides a mechanism for grouping logical data together.  
**SOAP Body:**This element contains the actual SOAP message for communication with the SOAP receiver; a message can contain zero or more bodies. Any information intended to be exchanged when the message reaches the intended destination goes in the message body.  
  
  
  
**Question 46: What is Duplex in WCF? Explain also Message Exchange Pattern?**  
**Answer: Duplex**  
Duplex is a Two-Way Communication Process. In which, a consumer send a message to the service and the service sends a notification that the request processing has been done. It is just like that we send some command about the printing of some papers to the Printer machine. The printer machine start communication by making the connection with our machine, and start printing the paper.

1. Duplex service allows calling back some operation (function) on the client.
2. Duplex service also knows as Call Backs.
3. All Binding does not support duplex service.
4. Duplex communication is not standard. They are absolute Microsoft feature.
5. wsDualHttpBinding supports duplex communication over HTTP binding.
6. Duplex communication is possible over netTcpBinding and netNamedPipeBinding
7. [ServiceContract(CallbackContract = **typeof** (DuplexServiceCallback))]
8. **public** **interface** IMyService
9. {
10. [OperationContract(IsOneWay = **true**)] //one-way message
11. **void** AddName(**string** name);
12. [OperationContract] //request-response message
13. **string** GetName(**string** name);
14. }

**Message Exchange Patterns in WCF:**The Message Exchange Pattern (MEP) provides a way of communication between client and server. MEP is a beauty of WCF. A services do some task depend on us or send a response to our request. The communication between client and server are done in form of messages. When we send a message as request and get a message as response from the service. The WCF Services support three types of Message Exchange Pattern:

* Request-Response
* One-Way
* Duplex

Why we need the Message Exchange Pattern:

1. When we want the calling of the function to not take more time.
2. We just want to call the function and not want wait for the response.
3. When we want that function call is not blocking and at the same time we want something out of the function (response) for our application.
4. When we want a two-way communication mechanism.

**Question 47: What you understand by Fault Exception in WCF?  
Answer:** A Fault Contract is a way to handle an error/exception in WCF. In C# we can handle the error using try and catch blocks at the client side. The purpose of a Fault Contract is to handle an error by the service class and display in the client side. Whenever the client makes a call to a service class and an unexpected exception comes such as, SQL Server is not responding, or Divide by zero; in this case the service class throws an exception to the client using the Fault Contract.   
By default when we throw an exception from a service, it will not reach the client. WCF provides the option to handle and convey the error message to the client from a service using a SOAP Fault Contract.  
To support SOAP Faults, WCF provides the FaultException class. This class has two forms:

1. **FaultException:** to send an untyped fault back to the caller.
2. **FaultException<T>:** to send a typed fault data to the client. It is basically a Generic Type.

**Syntax***[OperationContract]  
[FaultContract(typeof(MyException))]  
string getDetails(int value);*  
The Fault Contract sample demonstrates how to communicate error information from a service to a client. The sample is based with some additional code added to the servie to convert an internal exception to a fault.   
**Question 48: What is the concept of tracelevel in trace listeners?  
Answer:** To enable the **wcf trace** we need to mention the trace source in configuration file like Web.config. Mostly and widely used trace source is **System.ServiceModel.** Another trace source which is mostly used for WCF Message logging is **System.ServiceModel.MessageLogging.**  
**WCF tracing** provides us different **wcf tracing levels** as given below:

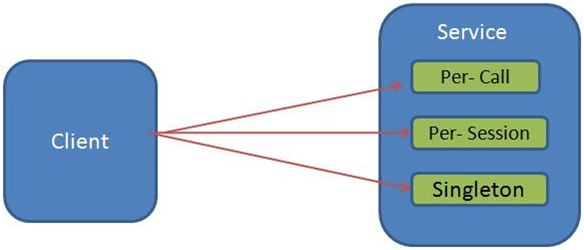
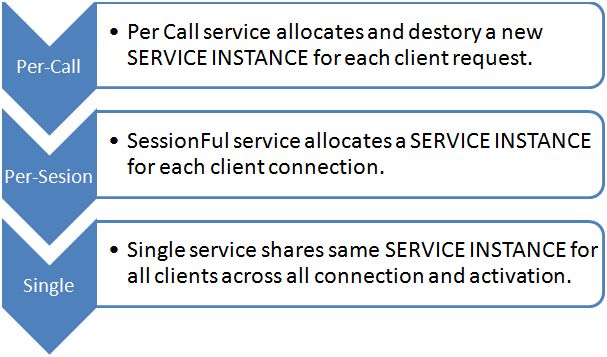
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Trace Level** | **Event Category** | **Tracing Details** | **Traced Events** |
| 1 | Off | - | – (No Tracing) | Ignores tracing. |
| 2 | Critical | Negative Events | Unexpected events or any error condition. When occurs, application may shut shortly or immediately. | All unexpected events OutOfMemoryException ThreadAbortException StackOverFlowException ConfigurationErrorsException SEHException Application start errors System Hang errors All other traces that cause application to terminate |
| 3 | Error | Negative Events | Unexpected events or any error condition. When occurs, application does not shut, it remains up & continues processing. | Traces all exceptions. |
| 4 | Warning | Negative Events | Indicates possibility of problems. Application remains up and continues the processing. | Traces all possible problems that are occurred or may occur. 1) TimeOut has exceeded 2) Rejected credentials 3) Receiving more request than expected. 4) Receiving Queue is nearby maximum capacity. |
| 5 | Information | Positive Events | Only informative traces that tells the successful milestone of application execution. | Traces all diagnostic data. |
| 6 | Verbose | Positive Events | Same as “Information”, however only difference is – it allows low level tracing. | Mostly used for Debugging purpose. |
| 7 | ActivityTracing | - | The traces are related to communication between components and activities. | - |
| 8 | All | Positive & Negative | It includes all above events | It includes all above events |

**Question 49: What is Instance Management in WCF?**AnswerInstance Management is the name for a set of techniques used by WCF to bind client requests to service instances, governing which service instance handles which client request. This is a service side implementation detail.  
In WCF, Instance Management is a technique to decide which service instance will serve a client request. It also decides when a client request will be served by a service instance. Instance Management is a very important factor when designing a service. Its affects:

* Scalability of Service
* Performance of Service
* Durability of Service
* Transactions
* Service Queues

Instance Management can be seen as a set of techniques to decide which service instance will serve a client request and when.   
Basically there are three instance modes in WCF:

* [Per-Call instance mode](http://wcftutorial.net/Per-Call-Service.aspx)
* [Per-Session instance mode](http://wcftutorial.net/Per-Session-Service.aspx)
* [Singleton Instance Mode](http://wcftutorial.net/Singleton-Service.aspx)

**Why we need Instance Management**we need Instance Management because applications differ too much in their needs for scalability, performance, throughput, transactions and queued calls. So one solution for all applications doesn't fit.  
  


* [3 Techniques For Instance Management in WCF](http://www.c-sharpcorner.com/UploadFile/81a718/3-techniques-for-instance-management-in-wcf/)

**Question 50: What is Information cards in WCF?  
Answer:** As we are using DerivativesCalculator sample to enhance the WCF application hence we can use it in very interesting way to add an Information Card to a WCF application. It provides more benefits than CardSpace.  
Benefits of adding Information Card:

* Authentication
* Authorization
* Reduce IT pain
* Granting trust to domains
* Helps in maintaining security

Information cards are virtual representations of a person's identity that are assured by a particular party. Information cards are analogous to real-world identity cards such as passports, driver's licenses, credit cards, and employee ID cards.  
**Sample Code:** To show the usage of DerivativesCalculator which help in adding Information card which require no external reference to do so.

1. namespace DerivativesCalculator
2. {
3. **public** **class** Calculator
4. {
5. **public** **decimal** CalculateDerivative(**string**[] symbols, **string**[] parameters, **string**[] functions)
6. {
7. **return** (**decimal**)(System.DateTime.Now > Millisecond);
8. }
9. }
10. }

* Information cards are more flexible than simple user names and passwords.
* Information cards employ strong cryptography, which makes their use more secure than passwords.
* Information cards can potentially present any type of identity claim that makes sense to all of the interacting parties and which users are willing to release.