WCF Concepts

1. System.ServiceModel (Assembly and namespace)
2. Service Contracts
3. Data Contracts
4. Service Classes
5. Service Host
6. Service Configuration
7. Service End Points
8. Bindings
9. Service Meta Data
10. Client Proxy Class
11. Client End Points

1) System.ServiceModel

- It is the namespace available at System.ServiceModel assembly.
- It provides necessary classes, to develop Service-oriented applications using WCF.
- This is required both in service and client side too.

2) Service Contracts

- Interfaces that define service operations are called as "Service Contracts".
- Service Interfaces must be decorated with [ServiceContract]
- Service Operations decorated with [OperationContract]

3) Data Contracts

- A class which defines the structure of data that is to be received and / or returned by a service is called as "Data Contract".
- Not necessary, if the operations receive and return simple types.
- Data Contract classes must be decorated with [DataContract]
- Its properties must be decorated with [DataMember]

Data Contracts (contd…)

- Must be serialized by using "DataContractSerializer", which is available at "System.Runtime.Serialization".
- The DataContractSerializer converts your DataContract class as XML Schema at run time.

4) Service Classes

- A class, which implements the Service Contract is called as "Service Classes".
- These are the classes that provide the service implementation.
- Implementation include with define actual functionality of operations.
- Can implement one or more service contracts.

5) Service Host

- ➢ Service Host is a listener application, which can listen the requests, coming from the clients.
- ➢ A WCF Service can run based on a Service Host only.
- ➢ In WCF, there are many ways to Host a Service.
  1. IIS Hosting (HTTP only)
  2. Self Hosting
     - Can be a Console App / Windows Forms App / WPF App etc.
  3. Windows Service Hosting
  4. WAS (Windows Activation Service) Hosting

Service Host (contd…)

- The following table shows you options for hosting WCF Services.

Sl. No	Hosting Tool	Features
1	Self Hosting	• Can be any .NET Application (Console Application / Windows Forms App / WPF Application etc.) • Supports **any** protocol. • While the application is running, your service will be alive.
2	IIS	• IIS can be installed on Windows o/s. • IIS will be started automatically with o/s. • Supports **only HTTP** protocol. • While the IIS is running, your service will be alive.
3	Windows Service	• Developed and installed on Windows o/s. • Windows Services can be started automatically with o/s. • Supports **any** protocol. • While the windows service is running, your wcf service will be alive.
4	WAS	• WAS can be Installed on Windows o/s. • WAS is shipped with Windows Vista / 7 / 2008 Server / 8 • WAS will be started automatically with o/s. • Supports **any** protocol. • While the WAS is running, your service will be alive.

The System.ServiceModel.ServiceHost class

- System.ServiceModel.ServiceHost is a class, which is used to activate a WCF Service.
- This must be instantiated in a ServiceHost application.
- It receives type of the Service class, as argument in its constructor.

6) Service Configuration

- Service Configuration provides more details about your WCF Service, to the "Service run time".
- It can be either declarative / dynamic.
- It can be utilized by Service Host, at run time.

```
<system.ServiceModel>
  <services>
      <service name="type of your service">
          {end point goes here…}
      </service>
  </services>
</system.ServiceModel>
```

7) Service End Points

- An EndPoint is ABC of WCF.

A. **Address (Where):** Defines where the clients can find your service. Defines URL of the service.

B. **Binding (How):** Defines the transport protocol.

C. **Contract (What):** Defines the name of your service contract.

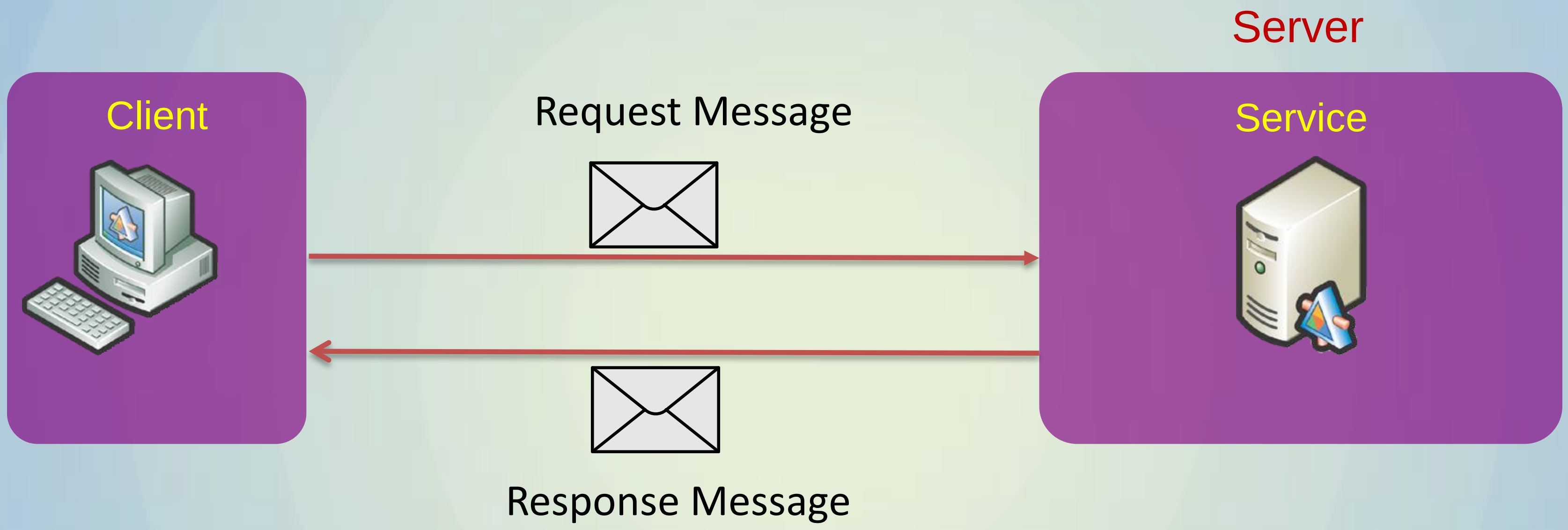
```
<endpoint
address="your service address"
binding="your binding name"
contract="your service contract name" />
```

Service End Points (contd…)

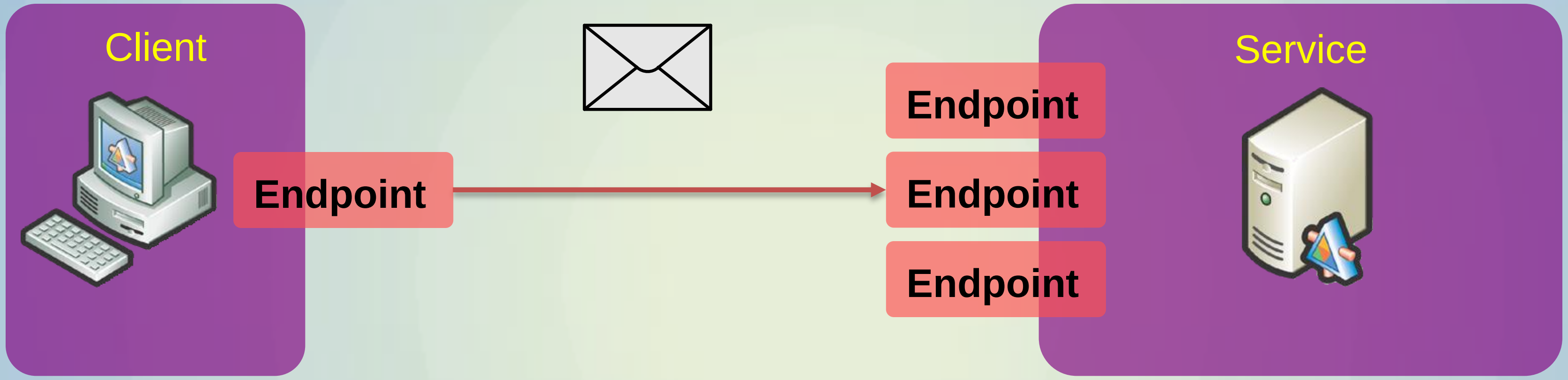
- A service can have one or more end points; but minimum is one.
- EndPoints can be defined declaratively / dynamically.
- At run time, the Service Host will load these end points.

```
<endpoint
address="your service address"
binding="your binding name"
contract="your service contract name" />
```

Clients and Services



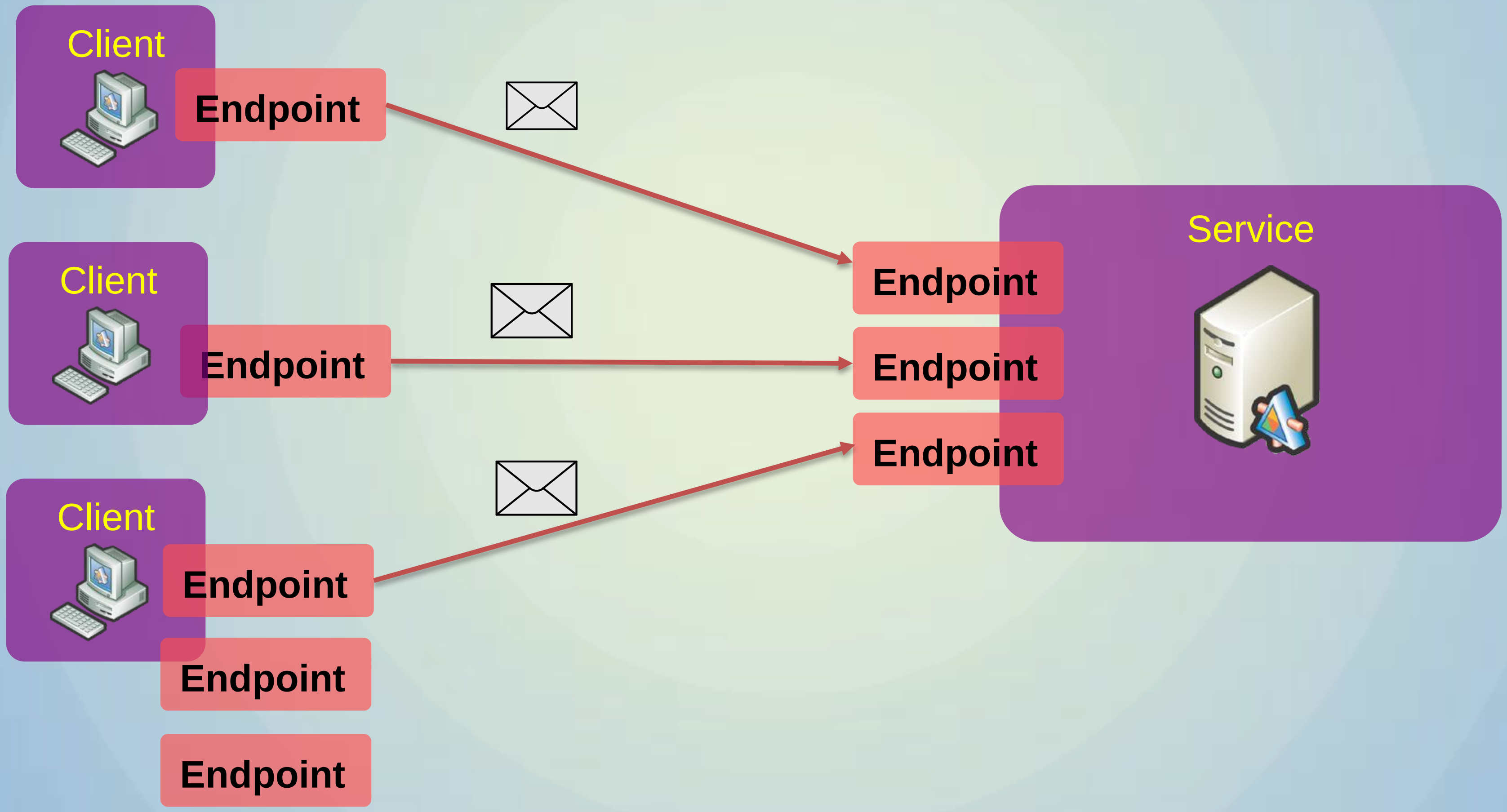
Endpoints



Why should I have an end point?
- To configure a protocol, for your service.

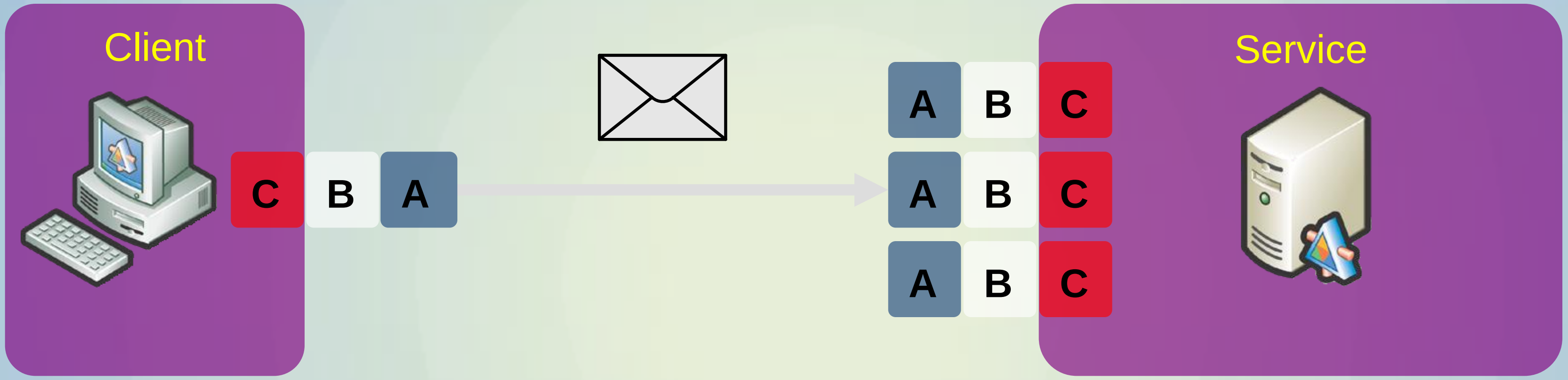
Note: At any moment, the exact interaction will be between one end point of client and another end point of service with same a, b, c.

Endpoints


Client
Endpoint
Client
Endpoint
Client
Endpoint
Endpoint
Endpoint
Service
Endpoint
Endpoint
Endpoint

Architecture of WCF

(Address, Binding, Contract)



Address

Where?

Binding

How?

Contract

What?

Address

- **Address:** An address is URL of the service.
- Ex:

```
http://microsoft.com:80/OrderService/WS
https://microsoft.com:443/OrderService/BP
net.tcp://microsoft.com:808/OrderService/TCP
net.pipe://microsoft.com/OrderService/NP
```

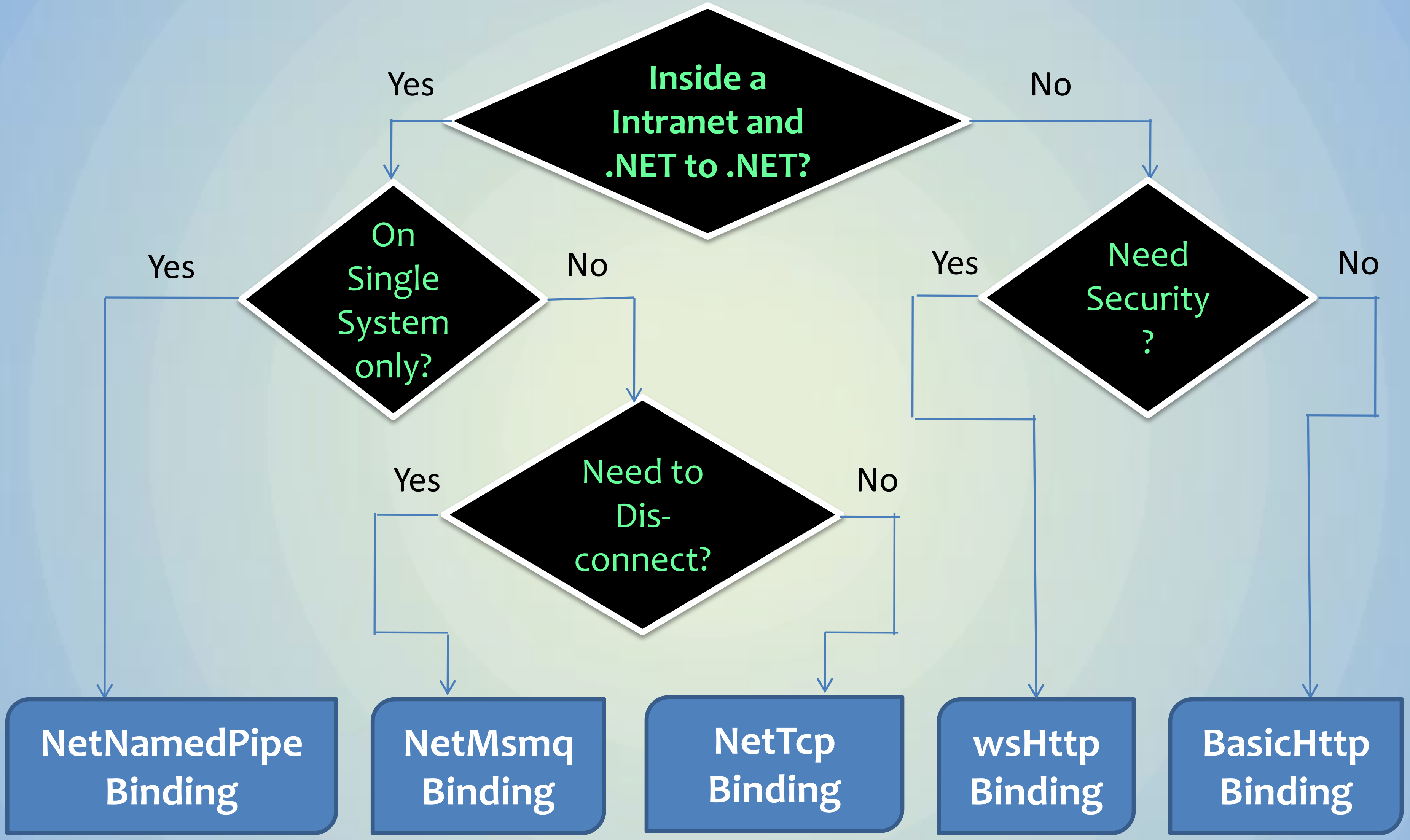
- Address in Service End Point specifies that, the service is available at this URL, and the clients can start communicating with the service using this URL.

8) Binding

- A Binding is the Communication Protocol that you want to use.
- Both Service and Client must use same Binding.
- Ex:
  - basicHttpBinding
  - wsHttpBinding
  - netTcpBinding

  etc.

Choosing a Binding



9) Service Meta Data

- Meta Data means Data about Data
- The Service Meta Data is the WSDL code, which describes about the Service, its operations and its arguments.
- WSDL is the acronym for "Web Service Descriptive Language".
- "WCF Run time" automatically generates the meta data of your WCF Service.
- The client utilizes this Meta Data, while adding Service Reference at client application.

10) Client Proxy Class

- It is a class, which contains methods same as the operation contracts; and those methods are used to call the service methods.
- In simple words, when you call a Client Proxy's method, a remote method gets called at server.

Client Proxy Class (contd…)

- Client Proxy class is
  1. Inherits from **System.ServiceModel.ClientBase<T>**
     - T is your service contract.
  2. Implements your service contract.

- Its methods make similar calls to methods of **Channel**.

11) Client End Points

- At client side also end points are must.
- A Client EndPoint is also ABC of WCF.

A. **Address:** Defines address of the service to call. (it must be same as your service address)

B. **Binding:** Defines the transport protocol, used for communication. (it must be same as your service binding)

C. **Contract:** Defines the name of your service contract. (it must be same as your service’s service contract)

Client End Points (contd…)

- A client can have one or more end points, that refer to same service; but at run time you can use only one.
- Client EndPoints also can be defined declaratively / dynamically.

```
<endpoint
address="your service address"
binding="your binding name"
contract="your service contract name" />
```