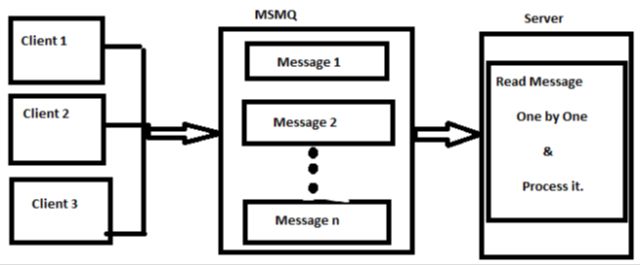
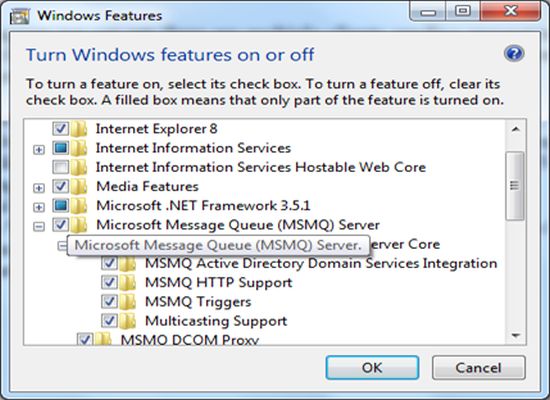
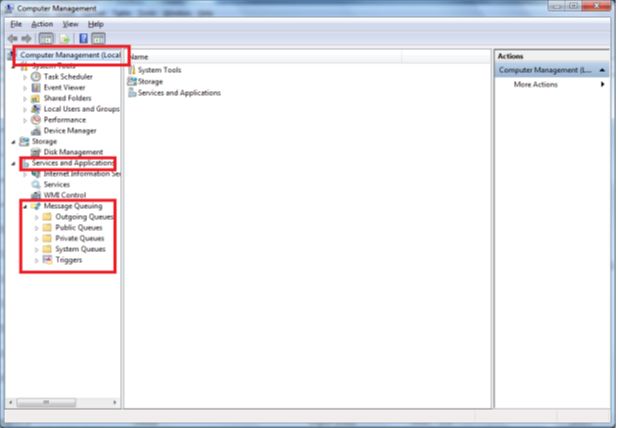
In this article we try to learn how to implement a MSMQ feature in a WCF service. The MSMQ feature allows us to create a program that can reliably send or receive messages over a wire without any loss of message. It makes use of a queue to store messages in the order in which they arrive. The queue is a First In First Out stack i.e. the message that arrived first will be the first that to be removed from the queue for processing by the server. MSMQ can be implemented in the manner that there are multiple clients sending a request to the server but because of overload, the server is busy processing previous client requests so in such cases MSMQ can be implemented to maintain a queue of requests from new clients and whenever the server becomes free it will take requests from the queue and process them one by one. By implementing MSMQ there are less chances of losing the message sent by the client.  
  
  
  
The following diagram depicts that.  
  
From the diagram above we can see there are multiple clients sending requests to the server, but if the server is busy then the requests are stored in the MSMQ queue and later when the server becomes free it proceeds to read and process messages one by one.  
  
Now to get to our topic of how to implement MSMQ in a WCF service, well first of all to implement the MSMQ feature we need to install the MSMQ server, because it's an optional component in your operating system, like the way we install IIS, we can use the same way here. For installing MSMQ use the following procedure.

1. Open the "Control Panel" -> "Programs and features"
2. Select "Turn Windows Features On or Off" from the left hand pane of the "Programs and Features" window
3. The Windows features dialog box will appear, from there select MSMQ Server (select all dependent checkboxes to install the full MSMQ server)  
     
   
4. Finally click on Ok to install it.
5. After installing it you can verify it by clicking on "Administrative Tools" inside "Control Panel"
6. Select "Computer Management" then expand "Services and Applications" and you will find Message Queuing with some sub folders inside it. Like the following:  
     
   

Until now we have installed MSMQ but now to work with it we must create a Queue that will store our messages if the server is not available, the queue can be created either manually or programmatically. In our case we will try to create a queue programmatically. Queues are of three types they are either public, private or System. For more information about that kindly search it.  
  
Ok now moving towards our program in this we have created a service contract that contains one operation contract with the name "SayYourWish" that accepts the name of the clients and their wishes. To maintain reliable messaging we have implemented the MSMQ feature in our service so that no client's wish is lost while transferring or due to server load or server unavailability.   
  
Here is my Service Contract that I defined.  
  
**IWish.cs**  
  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.ServiceModel;

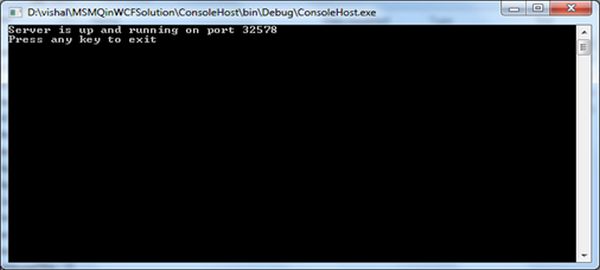
namespace WishListComponent  
{  
    [ServiceContract]  
    public interface IWish  
    {  
//IsOneWay=true denotes that there will no return message (or two way communication)   
//from the server to the client.  
        [OperationContract(IsOneWay = true)]  
        void SayYourWish(string wisherName, string yourWish);  
    }  
}  
  
**MyWish.cs**  
  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;

namespace WishListComponent  
{  
    public class MyWish : IWish  
    {  
            //This function will just display the name of the wisher and their wish on the Console          //Screen  
        public void SayYourWish(string wisherName, string yourWish)  
        {  
            Console.WriteLine("Client (Wisher): " + wisherName + " wish is :" + yourWish);  
        }  
    }  
}  
  
**ConsoleHost.cs**  
  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Messaging;  
using System.ServiceModel;  
using WishListComponent;

namespace ConsoleHost  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            using (ServiceHost host = new ServiceHost(typeof(MyWish)))  
            {  
                //Path of the Queue. Here we are creating a private queue with the name VishalQ   
                //where all our message would be stored, if server is unavialable.  
                string myQueue = ".\\private$\\VishalQ";  
                //Checking whether the queue exists or not.  
                if (!MessageQueue.Exists(myQueue))  
                {  
                    //If the queue doesnot exists it will create a queue with the name VishalQ  
                    //the second parameter false denotes that the queue would be a non transaction queue. If you want that your queue to be   
                    //transaction then make the second parameter to true.  
                    MessageQueue.Create(myQueue, false);  
                }  
                //finally opening the host to server the clients.  
                host.Open();  
                Console.WriteLine("Server is up and running on port 32578");  
                Console.WriteLine("Press any key to exit");  
                Console.ReadKey();  
            }  
        }  
    }  
}  
  
**App.Config file**  
  
<?xml version="1.0" encoding="utf-8" ?><configuration>  
  <system.serviceModel>  
    <diagnostics performanceCounters="All" />  
    <services>  
      <service name="WishListComponent.MyWish" behaviorConfiguration="myBehavior">

        <!--Address attribute specifies the name of the MSMQ Queue.-->  
        <endpoint name="wish" address="VishalQ" binding="netMsmqBinding"  
                                                                          bindingConfiguration="myMSMQ"contract="WishListComponent.IWish"/>  
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>  
        <host>  
          <baseAddresses>  
            <add baseAddress="net.msmq://localhost/private/"/>  
            <!--Both Mex and HttpBinding uses http://localhost:8888 port-->  
            <add baseAddress="http://localhost:32578"/>  
          </baseAddresses>  
        </host>  
      </service>  
    </services>  
    <bindings>  
      <!--The property exactlyOnce=false means that i am using non transactional queue. The property is by default true.-->  
      <netMsmqBinding>  
        <binding name="myMSMQ" exactlyOnce="false" >  
          <!--  
                                                                        If we donot set the security mode to none then the following error occurs.  
                                                                        Binding validation failed because the binding's MsmqAuthenticationMode property is set to   
                                                                        WindowsDomain but MSMQ is installed with Active Directory integration disabled.   
                                                                        The channel factory or service host cannot be opened.  
                                                            -->  
          <security mode="None"/>  
        </binding>  
      </netMsmqBinding>  
    </bindings>  
    <behaviors>  
      <serviceBehaviors>  
        <behavior name="myBehavior">  
          <serviceMetadata httpGetEnabled="true"/>  
          <!--This is for enabling an exception-->  
          <serviceDebug includeExceptionDetailInFaults="true"/>  
        </behavior>  
      </serviceBehaviors>  
    </behaviors>  
  </system.serviceModel>  
</configuration>

**Note**

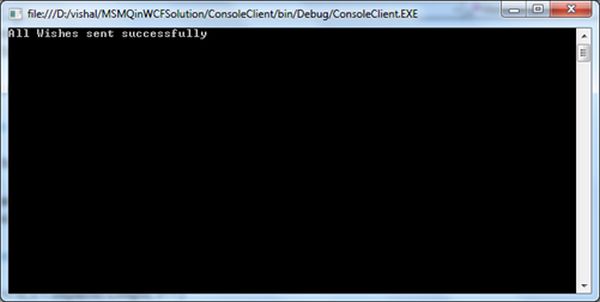
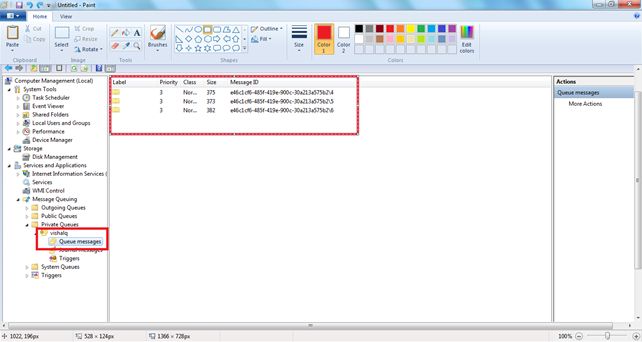
1. We will be making use of Self Hosting for hosting the program.
2. Our Client would be a Console Client.
3. For implementing MSMQ we will be making use of netMsmqBinding.
4. You need to add the service reference to the ConsoleClient Application, for adding it go to the bin folder of your ConsoleHost application through windows Explorer, from there double-click on the ConsoleHost.exe file so that it displays the following output.  
     
   
5. 5. Right-click on the ConsoleClient and click "Add Service Reference", in the service reference dialog box paste the URL as http://localhost:32578?wsdl that we specified in the App.config file for the mex endpoint.
6. Specify the name for your service as WishServiceRef and click on "Add".
7. Your service will be added to your ConsoleClient Application.

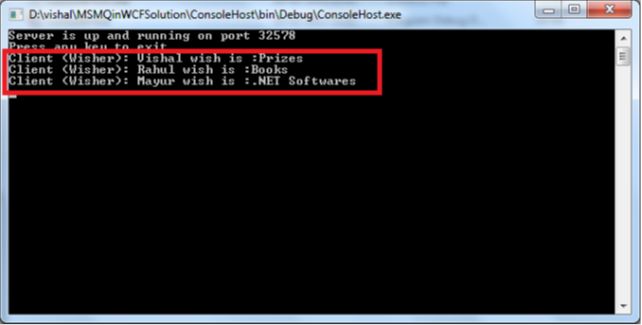
**ConsoleClient.cs**using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using ConsoleClient.WishServiceRef;

namespace ConsoleClient  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            string[] names = { "Vishal", "Rahul", "Mayur" };  
            string[] wishes = { "Prizes", "Books", ".NET Softwares" };

            WishClient[] objArray = new WishClient[names.Length];

            for (int i = 0; i < objArray.Length; i++)  
            {  
                objArray[i] = new WishClient();  
                objArray[i].SayYourWish(names[i], wishes[i]);  
            }  
             Console.WriteLine("All Wishes sent successfully");  
            Console.ReadLine();  
        }  
    }  
}  
  
Finally to Test the output. Do the following:

1. Close (to make the server unavailable) your ConsoleHost.exe if its already running
2. Run your ConsoleClient Program
3. It should display the following output:  
     
   
4. Close the ConsoleClient output screen. And now check your queue.
5. Go to "Control Panel" -> "Administrative Tools" -> "Computer Management"
6. Go to "Message Queuing" inside "Services and Applications"
7. Go to your queue in this case "VishalQ" inside "Private Queues", expand it and select "Queue Messages"
8. On the right hand pane you will see all your messages are stored inside the queue.
9. The following diagram depicts that.  
     
   

When we ran our ConsoleClient App at the time our Server i.e. the ConsoleHost was not running i.e. it was unavailable, now go to the bin folder of your ConsoleHost program through Windows Explorer and double-click the ConsoleHost.exe. You will find that the message from the queue will be removed and displayed on the ConsoleHost Screen like the following.  
  
  
  
Hope you liked the article, in my future article we will try to implement Duplex or two-way communication between the client and the service using MSMQ in WCF.