Many thanks to **Eben Roux** for code documentation and review.

Chapter 2 : NServiceBus Architecture

In this chapter, we will be focusing on the NServiceBus Architecture. We will discuss the different message and storage types supported in NSB. This discussion will include an introductory to some of the tools and advantages of using NSB as we conceptually look at how some of pieces fit together while backing up the discussions with code examples.

These were ran in VS2012 in Windows Server 2012, with MSMQ, DTC, NServiceBus references, and SQL Server 2012. If running in Windows 8.1, please run as "administrator", and there may be warnings if running the first time that some of the queues do not exist and it is creating the queues.

The Source code in this section:

In this section, we will be using the **TimeoutManager** solution with the following projects:

- TimeoutManager – This project will perform several timeout functions through NServiceBus.

If running in Windows 8.1, please run as "administrator", and there may be warnings if running the first time that some of the queues do not exist and it is creating the queues.

The Source code in this section:

In this section, we will be using the **MessageMutators** solution with the following projects:

- Client – The client will send messages to the server.
- Server – Will receive the mutated message.
- Messages – The message format being passed between client and server.
- MessageMutators – This project will contain the mutation code to compress and uncompress the messages in "TransportMessageCompressionMutator.cs" and validate the message annotation in "ValidateMessageMutator.cs".

The Server is setup to run by default in Visual Studio 2012, please run the Client in a separate instance of Visual Studio 2012.

The Source code in this section:

In this section, we will be using the **Encryption** solution with the following projects:

- Client – The client will send an encrypted credit card messages to the server.
- Server – The server will receive the credit card message and decrypt it.
- Messages – The message format being passed between client and server.

The Server is setup to run by default in Visual Studio 2012, please run the Client in a separate instance of Visual Studio 2012.

The Source code in this section:

In this section, we will be using the **ScaleOut** solution with the following projects:

- Orders.Messages – The common messages for the sender and handlers.

- Orders.Sender – Will send messages to Orders.Handler to be handled across the workers, worker1and worker2.
- Orders.Handler.Worker1 – One of the worker services that is using a worker profile to send a response back to the sender. This will be an additional worker copy of Orders.Handler.
- Orders.Handler.Worker2 – One of the worker services that is using a worker profile to send a response back to the sender. This will be an additional worker copy of Orders.Handler.
- Orders.Handler– an endpoint which processes the message and configured to the distributor. This will be the master profile that the sender will send the place order command to in the "orders.handler" MSMQ. In the Visual Studio 2012 debugger, the "NServiceBus.Integration NserviceBus.Master" is set in the command line to be used instead of "Configure.Instance.RunDistributor()".

You may need to run Visual Studio as an administrator.

- If one does not start up 'Orders.Handler' first and wait for it to be up-and-running the workers fail trying to access the distributor queue.

The Source code in this section:

In this section, we will be using the **ScaleOut-Performance** solution with the following projects:

- Orders.Messages – The common messages for the sender and handlers.
- Orders.Sender – Will send messages to Orders.Handler to be handled across the workers, worker1and worker2.
- Orders.Handler.Worker1 – One of the worker services that is using a worker profile to send a response back to the sender. This will be an additional worker copy of Orders.Handler.
- Orders.Handler.Worker2 – One of the worker services that is using a worker profile to send a response back to the sender. This will be an additional worker copy of Orders.Handler.
- Orders.Handler– an endpoint which processes the message and configured to the distributor. This will be the master profile that the sender will send the place order command to in the "orders.handler" MSMQ. In the Visual Studio 2012 debugger, the "NServiceBus.Integration NserviceBus.Master" is set in the command line to be used instead of "Configure.Instance.RunDistributor()".

You may need to run Visual Studio as an administrator.

- If one does not start up 'Orders.Handler' first and wait for it to be up-and-running the workers fail trying to access the distributor queue.

This is the same as the ScaleOut example, except that performance information will be set in the Worker projects, such as EndpointSLA in the EndpointConfig.cs.

The Source code in this section:

In this section, we will be using the **Gateway** solution,

1) Headquarter.Messages – The common messages for the Headquarters, SiteA, and SiteB.
2) Headquarter – Will receive messages from http://localhost:25899/Headquarter/ and http://localhost:25899/Headquarter2/, and send messages to http://localhost:25899/SiteA/ and http://localhost:25899/SiteB/ .

3) SiteA – A project that will receive update price information from Headquarters across http://localhost:25899/SiteA/ and respond that it was successful back to the Headquarters across http://localhost:25899/Headquarter2/.

4) SiteB – A project that will receive update price information from Headquarters across http://localhost:25899/SiteB/ .

5) WebClient – Will have a Index.htm page to send a JSON script to http://localhost:25899/Headquarter/ .

A "nservicebus" database must be present on the local SQLExpress.

Please click on the "index.htm" to run the web client.

The Source code in this section:

In this section, we will be using the **ConsoleReadTasks** solution, which is a task to notify through email of queues running, **so email, as an SMTP connection**, must be set up to run on the system locally, see book, and on Windows 8.1, you will need to run as "administrator". The book describes using http://smtp4dev.codeplex.com/ for testing SMTP.