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

Chapter 5 : The Persistent Architecture

For the ESB bus, persistence is the key element for the storing of messages, which could be associated as business objects that run through the ESB workflow. There are other persistent elements that comprise of the meta-data that define how the messages and workflow is being handled in the ESB through configuration. Persistence can also be considered the feedback that the ESB gives back to the system in form of logging, errors and audits. In this chapter, we will be covering persisting items to the database, including messages and logging.

The Source code in this section:

In this section, we will be using the **PayQueue** solution under the **PayQueueMSMQ** directory:

1) MyMessages – A payment message used for the projects.

2) AppForWritingXML – A projects that writes payment XML files to a local C: drive.

3) AppForReadingXML – A project for reading XML files from the drive, saves a copy to the local database using Entity Frameworks using the routines from the AppForWritingToTable routines, and sending them to MSMQ to process as messages.

4) AppForWritingTables – Just the Data Access routines for AppForReadingXML.

First, Run **"AppForWritingXML"** to create XML files, and then "**AppForReadingXML**" to prepare the SQL tables and MSMQ using NServiceBus.

The **"AppForWritingXML"** creates 5 XML files into C:\Load_XML_Files.

The **AppForReadingXML** will load the XML messages into the Payment table. The Payment table for sending payments, the AppForReadingXML table for receiving payments, the unicastconfig table and auditconfig tables need to be created. We see that the messages move from files to the Payment table to the AppForReadinXML table using various coding methods.

A "PayQueue" Database must be created in the SQLExpress, with a Payment and AppForReadingXML tables created with the proper fields.

If these tables are not present, then run the Model.edmx to create the tables from the AppForWritingForTables's Model.edmx using the "Generate Database from model". This will create a file called "Model1.edmx.sql" that when run will create the tables.

These were ran in VS2012 in Windows Server 2012, with MSMQ, DTC, NServiceBus references, and SQL Server 2012 Express LocalDB installed.

Ensure that DTC, MSMQ, and NServiceBus is set up per http://docs.particular.net/nservicebus/preparing-your-machine-to-run-nservicebus .

In this section, we will be using the **NserviceBus.SqlServer.Samples-master\VideoStore.SqlServer** which is described in SQL Queueing Sample. The sample runs a VideoStore for SQL queueing for ordering videos.

The solution was ran in VS2012 in Windows Server 2012, with MSMQ, DTC, NServiceBus references, and SQL Server 2012 Express LocalDB installed.