# Message Queue Fundamentals in .NET

**Module 8 Setup Instructions**

In Module 8 we implement the IMessageQueue interface using Azure and AWS for cloud messaging.

## Pre-requisites

The demo solution is delivered in Visual Studio 2013.

The solution uses NuGet package restore to load packages during the build.

To enable this in Visual Studio, open *Tools…Library Package Manager…Package Manager Settings* and ensure both options (*Allow NuGet to download missing packages* and *Automatically check for missing packages during build in Visual Studio*) are ticked.

The web project is set up to use IIS (not IIS Express), so you will need IIS installed, or you can change the configuration for the web project.

The database uses SQL Server (Express edition is fine) – the connection strings in App.config expect a default unnamed instance on the local machine

## Running the Queries

For Azure:

* In your Service Bus namespace, create queues called:
  + **doesuserexist**
  + **unsubscribe**
* In your Service Bus namespace, create a topic called **unsubscribed-event** with subscriptions called:
  + **crm**
  + **fulfilment**
  + **legacy**
* View the Connection Information for the namespace to get your SAS connection string
* Replace the text “YOUR-CONNECTION-STRING” ServiceBusMessageQueue.cs with your Azure connection string
* In MessageQueueFactory.cs ensure the ServiceBusMessage queue is returned in the Create method

For AWS:

* Create new SQS queues called:
  + **doesuserexist**
  + **unsubscribe**
  + **unsubscribe-crm**
  + **unsubscribe-fulfilment**
  + **unsubscribe-legacy**
* In SNS, create a topic called **unsubscribed-event**,subscribe the following queues to the topic and be sure to flow “Raw message delivery” for each queue:
  + **unsubscribe-crm**
  + **unsubscribe-fulfilment**
  + **unsubscribe-legacy**
* Navigate to Security Credentials (from the drop-down with your account name in the AWS Management Console), and get your Access Key and Secret Key
* Copy the access key and secret key into AwsMessageQueue.cs, replacing “YOUR-ACCESS-KEY”, “YOUR-SECRET-KEY” and “YOUR-URL” values
* Copy the SQS queue URLs and SNS topic ARN into AwsMessageQueue.cs, replacing the relevant URLs and ARN
* In MessageQueueFactory.cs ensure the AwsMessageQueuequeue is returned in the Create method

## Running the Solution

Build the solution and navigate to **http://localhost/Sixeyed.MessageQueue.Web/unsubscribe** in a browser window to submit the form and start the unsubscribe workflow.

In the **before** solution, the website will use ZeroMQ for messaging.

In the **after** solution the website will use either Azure or AWS for messaging, depending on the return type from the MessageQueueFactory.