# Message Queue Fundamentals in .NET

**Module 10 Setup Instructions**

In Module 10 we implement the IMessageQueue interface using WebSphere MQ, and run a final version of the solution using different queue technologies for different parts of the unsubscribe workflow.

## Pre-requisites

We use all the on-premise queue technologies from the course in this module’s demos, so you will need to have followed the setup instructions for all other modules, except the cloud modules – 7 & 8.

Additionally you will need to create the following entities in your WebSphere MQ server instance, under the **SC.UNSUB** Queue Manager:

* Local queues (created with the default settings are fine) named:
  + **doesuserexist**
  + **unsubscribe**
  + **unsubscribed.crm**
  + **unsubscribed.fulfilment**
  + **unsubscribed.legacy**
* Model queues:
  + **dynamic.response.model**
* Topics
  + **unsubscribed.event** – with a topic string unsubscribed
* Subscriptions, linking the **unsubscribed.event** topic to the destination queue having the same name as the subscription, with a wildcard topic string of **#**:
  + **unsubscribed.crm**
  + **unsubscribed.fulfilment**
  + **unsubscribed.legacy**

## Running the Solution

The *before* folder has the demo solution with the lo4net and Topshelf integration, but without the WebSphere MQ implementation. This version uses MSMQ for all messaging apart from the request-response pattern, which is processed through ZeroMQ.

The *after* folder contains the demo solution with the complete WebSphere MQ implementation, configured to use ZeroMQ for request-response, MSMQ for fire-and-forget and WebSphere MQ for publish-subscribe.

With both solutions, you can run the end-to-end tests and they should pass, using the configured queue implementations. Start all instances of the message handler using the *StartHandlers.cmd* batch file and navigate to **http://localhost/Sixeyed.MessageQueue.Web/Unsubscribe** to test the functionality.