Course 2957B Module 8 – additional notes

Exceptions in Asynchronous methods

In this module we looked at making asynchronous calls. If a delegate is invoked using **BeginInvoke** it executes on a worker thread. When execution is complete, the return value can be obtained using **EndInvoke**.

What happens if the delegate executing in the worker thread causes an exception? If **BeginInvoke** throws an exception, then you'll know that the asynchronous method has not been called. **EndInvoke** is used to harvest the results and allow the system to perform cleanup. If the delegate method throws an exception, then the exception is *thrown again* in the thread where your code calls **EndInvoke**. You can catch the exception by wrapping the call to **EndInvoke** in a try-catch block, like this:

```
try
{
     d.data = add.EndInvoke(iar);
}
catch (Exception e) { Console.WriteLine(e.Message); }
```

Other asynchronous calls behave in the same way. For example if you call the **BeginRead** method of a **FileStream**, and an exception occurs during the read process (for example a disk failure) then the exception is re-thrown and can be caught where **EndRead** is called.

Examples: AsyncPollingDemo and AsyncCallbackDemo projects

To demonstrate this in the example code, the delegate method, **Add**, can throw an exception.

- In **AsyncPollingDemo**, **EndInvoke** is called in the main thread so the exception can be caught in the main thread.
- In **AsyncCallbackDemo**, **EndInvoke** is called in the **Callback** method which runs on the worker thread, so the exception can be caught in the worker thread.

To observe the exception behaviour, remove the exception handling code in one of these examples. Start debugging. Execution will halt at the point in the **Add** method where the exception is first thrown. You can view detail of the exception and look in particular at the Stack Trace. If you now choose to continue, execution will again halt at the call to **EndInvoke**. Again view detail and examine the Stack Trace. You will see that the trace indicates that the exception has been re-thrown.