

# LeahCahillReport2

*by* Leah Cahill

---

|                |                                                          |                 |       |
|----------------|----------------------------------------------------------|-----------------|-------|
| FILE           | 2254_LEAH_CAHILL_LEAHCAHILLREPORT2_113620_177478247.DOCX |                 |       |
| TIME SUBMITTED | 04-MAY-2018 03:08PM (UTC+0100)                           | WORD COUNT      | 1686  |
| SUBMISSION ID  | 88388500                                                 | CHARACTER COUNT | 10235 |

# Retailer WCF Report

DISTRIBUTED ENTERPRISE APPLICATIONS

LEAH CAHILL (P140092)

## Contents

|                                                               |   |
|---------------------------------------------------------------|---|
| Analysing the design of the software components .....         | 2 |
| Setting up the P2P exchange pattern. ....                     | 2 |
| The Client (interface of application). ....                   | 2 |
| Encryption in the application.....                            | 4 |
| Connecting to a Database to store Users data .....            | 5 |
| Threads and Delegates .....                                   | 6 |
| Deploying the application as part of a real-life system ..... | 7 |
| The design.....                                               | 8 |
| References .....                                              | 9 |
| <br>Figure 1 .....                                            | 3 |
| Figure 2 .....                                                | 4 |
| Figure 3 .....                                                | 4 |

## Analysing the design of the software components

### Setting up the P2P exchange pattern.

The small e-commerce system created for this assignment was a peer-to-peer application. A peer-to-peer network consists of nodes, these nodes are the servers that are connected to the network [1]. To begin this type of exchange pattern, the binding must be set in the configuration file. Bindings in WCF define how the client and the service communicates.

The binding set in this application is:

```
<bindings>
  <netPeerTcpBinding>
    <binding name="storeBinding">
      <security mode="None" />
    </binding>
  </netPeerTcpBinding>
</bindings>
```

And then added into the endpoint in the configuration file as:

```
<client>
  <endpoint name="storeEP"
    address="net.p2p://localhost:8000/Store"
    binding="netPeerTcpBinding"
    bindingConfiguration="storeBinding"
    contract="LeahCahill_AssignmentTwo_WCF.I
  </client>
```

This binding is important as it enables nodes to host multiple peer channels and is also the B in the ABC's of WCF (Address, Binding and Contract) [2]. It is also important that the security mode is set, it is set to none here.

The service contract is another concept of WCF. This contract defines what kind of message exchange pattern is going to take place when the system is running. The service contract implemented in this P2P application is in the store.cs file, it is set with `CallbackContract = typeof(IItem)`, this is what is known as the callback method. If the contract is set to a duplex contract, then this gets/sets the type of callback contract.

The operation contracts within this service contract is set to `[OperationContract(IsOneWay = true)]` which means that the contract for this method is set as a one-way and has no return value. You set the value to no return by `"= true"`.

The service behaviour set in the store.cs class is created with the instance context mode: `[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]`. As this mode is currently set to single, this means that only a single instance is created and handles all requests [3].

### The Client (interface of application).

The interface for this application is a windows form application.

In the class that is responsible for the form and where the component is initialized. It contains this code:

```
public Form1()
{
    InitializeComponent();

    store s1 = new store(this);

    DuplexChannelFactory<IItem> it = new DuplexChannelFactory<IItem>(new InstanceContext(s1), "storeEP");
    channel = it.CreateChannel();
    ((ICommunicationObject)channel).Open();
}
```

This part of the code is where it creates a new instance of a duplex channel factory, and `ICommunicationObject` is where it defines the contract for the basic state machine for all communication-oriented objects in the system. As mentioned, the Instance context mode is set to single in the class store, this is where it is initialized once. The channel is created here with this code, and it is created as a self-hosted application (There is no host application).

Regarding this windows form application interface, the buttons and/or text boxes are added onto the form design and then double clicked so behaviour can be implemented in the form code. For example, the "Add Book" button code looks like this:

```
1 reference
private void addBookButton_Click(object sender, EventArgs e)
{
    channel.SendItem1("book");
    channel.Item1Price(10);
}
```

Which corresponds to this:

```
public void AddPrice1(string prc)
{
    if (prcList.InvokeRequired)
    {
        //We are not on the thread which created this control.
        //Must invoke an event to update it.
        prcList.Invoke(new UpdateListBoxDelegate(AddPrice1), prc);
    }
    else
    {
        //We are on the thread which created this control.
        //We can just update it directly.
        prcList.Items.Add(10);
    }
}

public void AddItem(string itm)
{
    if (itemListBox.InvokeRequired)
    {
        //We are not on the thread which created this control.
        //Must invoke an event to update it.
        itemListBox.Invoke(new UpdateListBoxDelegate(AddItem), itm);
    }
    else
    {
        //We are on the thread which created this control.
        //We can just update it directly.
        itemListBox.Items.Add("book");
    }
}
```

This ensures that we are on the thread and it is updated when this button is clicked. When the application is running, the user can click on the button "Add Book" and the word "book" will appear on the list of items to the right and the price of £10 will show next to it. Shown in Figure 1.

Figure 1

## Encryption in the application

This application requires users to log in if they wish to purchase an item. This requires them to sign up and the username and password is inserted into a database. To include some type of security in this application, implementing encryption using an advanced encryption standard (AES) algorithm to store the password is an appropriate choice [4], there are several other data encryption types such as Rivest-Shamir-Adleman encryption (RSA) [5]. The popular Rijndael algorithm is the method used here for the encryption of the password.

When the user enters their password, a try block is opened, and a new file stream is created so the data will be encrypted into a text file (called Output.txt). An instance of the Rijndael class is then created, which is the cryptography engine. A key is generated and then a CryptoStream is created which is passed to the file stream and encrypted. The data is passed to a stream writer which is created to write the data to the file:

```
//Create a StreamWriter to write to the file.
StreamWriter SWriter = new StreamWriter(CryptStream);

// data to be encrypted.
SWriter.WriteLine(passwordregistertext.Text);
```

and the handles are all closed when this is complete. Passwordregistertext, (shown in code example above) is the name of the password textbox in the form design.

### An example of using the encryption:

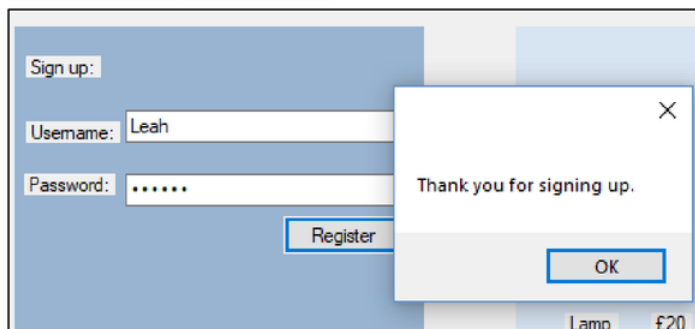


Figure 2: The user has entered the password and clicked register.

Figure 2

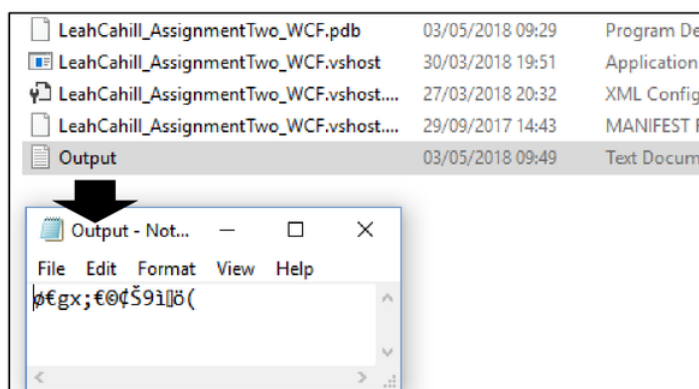


Figure 3: The file created to store the password is Output.txt and this is created upon user input. Here, this screenshot shows the Output.txt file open and the encrypted password.

Figure 3

## Connecting to a Database to store Users data

The application is connected to a local SQL database which enables the connection via a connection string, opening the connection and by adding the reference: System.Data.SqlClient;

```
private void Registerbutton_Click(object sender, EventArgs e) // registering user to the database
{
    string connectionString;
    SqlConnection cnn;
    connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=C:\USERS\LEAHC\DOCUMENTS\CUSTOMER.MDF;Integr
    cnn = new SqlConnection(connectionString);
    SqlCommand cmd;
    cnn.Open();
    string s = "insert into customer values(@p1,@p2)";
    cmd = new SqlCommand(s, cnn);

    cmd.Parameters.AddWithValue("@p1", textBoxUsername.Text);
    cmd.Parameters.AddWithValue("@p2", passwordregistertext.Text);
    cmd.CommandType = CommandType.Text;
    int i = cmd.ExecuteNonQuery();
    cnn.Close();

    try
    {
        //Create a new file to write our encrypted data to.
        FileStream fs = new FileStream("Output.txt", FileMode.Create);

        //Create a new instance of the RijndaelManaged class which will be our cryptography engine.
        RijndaelManaged cryptEngine = new RijndaelManaged();
    }
}
```

This is the part of the application that stores the user's data. If it were to become a real-life system, it would have major security flaws as it stores the password in plain text currently (as well as encryption into a file). This would have to be amended so it only stores the encrypted version. You can see the beginning of the Try block for the encryption in the screenshot above.

When the user wishes to log in, the code below shows how it searches the database to see if the user has registered:

```
1 reference
private void Loginbutton_Click(object sender, EventArgs e)
{
    string connectionString;
    SqlConnection cnn;
    string userlogin = textBoxLoginusername.Text;
    string userpw = passwordregistertext.Text;

    connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=C:\USERS\LEAHC\DOCUMENTS\CUSTOMER.MDF;Integr
    cnn = new SqlConnection(connectionString);
    SqlCommand cmd = new SqlCommand("SELECT * FROM customer WHERE username = '" + userlogin + "' AND [password] = '" + u
    cnn.Open();

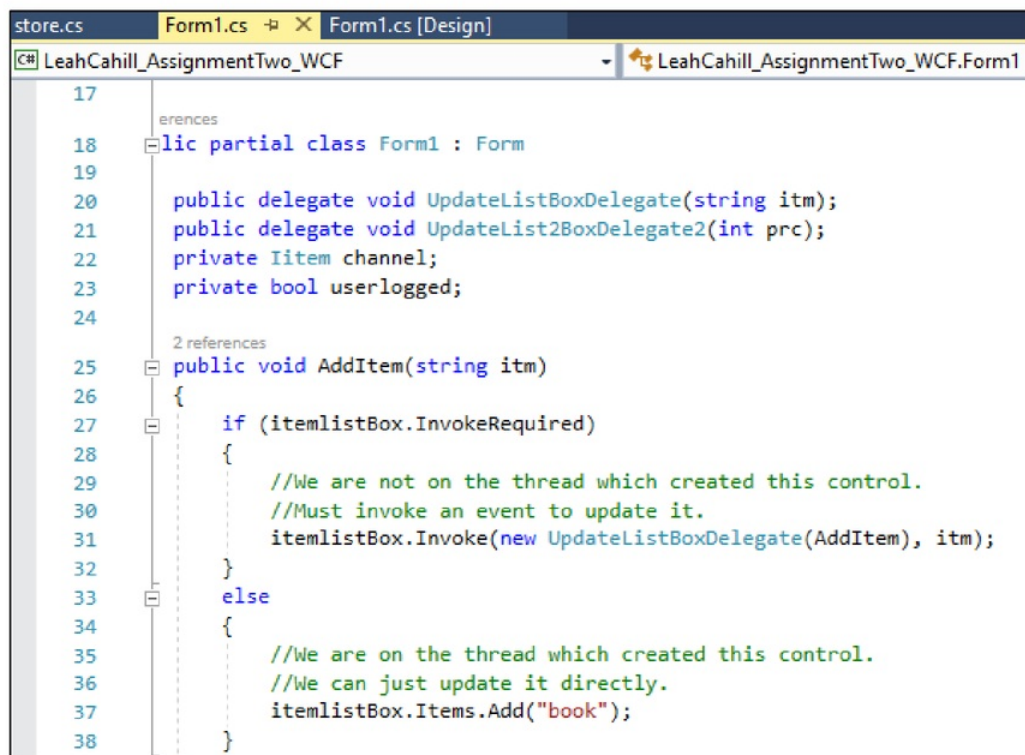
    SqlDataReader sdr = cmd.ExecuteReader();
    // If the record can be queried, it means passing verification, then open another form.
    if ((sdr.Read() == true))
    {
        MessageBox.Show("Thank you for logging in");
        //set user logged to true when they have logged in after registering
        userlogged = true;
    }
}
```

## Threads and Delegates

This P2P application created enables multiple instances to be built and run, this would represent an e-commerce website in a real-life situation where multiple customers will most likely visit at similar times.

The interactions of the user on the client side do not interfere each other and the buttons are linked to threads which is an entry point to start the execution of code.

These threads require a delegate and you can have more than one instance of the same delegate as they are abstract objects. Shown below in the screenshot of code, the delegate is declared at the top with "public delegate void..." and is instantiated in the AddItem method by itemListBox.Invoke(new UpdateListBoxDelegate(AddItem) [6]. When it is instantiated, it must include the same parameters and in the same order as the declared delegate.



```
17
18 public partial class Form1 : Form
19
20     public delegate void UpdateListBoxDelegate(string itm);
21     public delegate void UpdateList2BoxDelegate2(int prc);
22     private Iitem channel;
23     private bool userlogged;
24
25     2 references
26     public void AddItem(string itm)
27     {
28         if (itemListBox.InvokeRequired)
29         {
30             //We are not on the thread which created this control.
31             //Must invoke an event to update it.
32             itemListBox.Invoke(new UpdateListBoxDelegate(AddItem), itm);
33         }
34         else
35         {
36             //We are on the thread which created this control.
37             //We can just update it directly.
38             itemListBox.Items.Add("book");
39         }
40     }
```

## Deploying the application as part of a real-life system

Web services replace complex interfaces with a standard mechanism to communicate, this enables businesses to use web services for their e-commerce systems and link with customers/suppliers through the internet [7].

This application would need some changes before it could be deployed as part of a commercial distributed system. To begin with, the database stores the passwords that the users enter and stores the encrypted password. This would need to change so it does not store the password as plain text into the database, as it would not comply with the Data Protection Act (1998) where the companies must protect personal data [8].

There is also currently no payment method set up to enable the users to make a “real” purchase. The chosen payment methods could be with a separate company such as PayPal as PayPal has the security and is popular, so most customers will likely have an account. However, if the application decided to take payments from the customers debit cards, then the security of how that customers data must be considered. The payment details must not be stored. Expanding upon the security, measures should be put into place to regularly test that the network is secure and update regularity to avoid weaknesses in the network.

Using WCF for web services may have many advantages such as building reliable systems and ensuring that the data transmission between the system is efficient. WCF is good for building real-time applications which is what would be necessary for an e-commerce system for scenarios such as; enabling a user to purchase an item and for the system to check that the item is in stock at that very moment [9].

This application also calculates the postage incorrectly when it is applying the postage price on top of the subtotal of items. This would have to be fixed if it was deployed as part of a real system.

## The design

The application has been created with a simple interface. The sign-up section is in the top left, the log in section is below it. The items are in the middle with a “basket” that automatically shows what items you have clicked on when you select “Add”, underneath the basket, you are able to calculate the subtotal. To the right, it shows delivery charges that you can calculate and then a button to purchase.

The screenshot shows a web application interface with the following sections:

- Sign up:** Username and Password input fields, a Register button, and a Read Password back button.
- Already signed up? Log in:** Username and Password input fields, and a Log in button.
- Items:** A grid of items with their prices and an Add button for each:
  - Book £10
  - CD £15
  - Lamp £20
  - DVD £8
- Basket:** A table with two columns: Item and Price. Below the table are input fields for 'Items:' and a 'Calculate subtotal' button.
- Our Delivery Charges:** A table showing delivery charge ranges and costs:

| Range          | Charge |
|----------------|--------|
| < £50          | £6     |
| £50 - £99      | £4     |
| £100 - £149.99 | £2     |
| > £150         | Free   |

Below this table are buttons for 'Calculate Total with postage' and 'Total: £ ...'.
- Checkout:** A button at the bottom right.

Buttons have been added and the code implemented, so that the user must be logged in to check out. If the user has not logged in, an alert will appear informing the user to log in before it says, “Purchase Complete”.

A customer is able to add multiple items of the same item, shown below.

This screenshot shows the application after multiple items have been added to the basket. The 'Items' section remains the same. The 'Basket' section now displays the following items in its table:

| Item | Price |
|------|-------|
| book | 10    |
| CD   | 15    |
| DVD  | 8     |
| Lamp | 20    |
| DVD  | 8     |
| DVD  | 8     |

Below the table, the 'Items:' input field now contains the number '4'. The 'Calculate subtotal' button is still present. At the bottom right, the total price is displayed as '£ 53'.

## References

- [1] N. Pathak, Pro WCF 4: Practical Microsoft SOA Implementation, Appress, 2011.
- [2] M. L. Bustamante, Learning WCF: A Hands-on Guide, O'Reilly Media Inc., 2007.
- [3] A. Singh, "Instance Context Mode," C# Corner, 6 April 2015. [Online]. Available: <https://www.c-sharpcorner.com/UploadFile/de41d6/instance-context-mode-in-wcf/>. [Accessed 25 April 2018].
- [4] S. L. P. Charles P. Pfleeger, Security in Computing, Prentice Hall Professional, 2013.
- [5] S. M. Subijit Mondal, "Data security-modified AES algorithm and its applications," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 2, pp. 1-8, 2014.
- [6] "C# Delegates," Tutorialspoint, [Online]. Available: [https://www.tutorialspoint.com/csharp/csharp\\_delegates.htm](https://www.tutorialspoint.com/csharp/csharp_delegates.htm). [Accessed 3 May 2018].
- [7] R. L. G. Z. R. G. Mario Bravetti, "Web Services for E-commerce: guaranteeing security access and quality of service," *SAC '04 Proceedings of the 2004 ACM symposium on Applied computing*, pp. 800-806, 2004.
- [8] "Data Protection," Gov.UK, [Online]. Available: <https://www.gov.uk/data-protection>. [Accessed 3 May 2018].
- [9] "Top 10 Differences Between WCF and Web Services," BrainVire, 15 April 2015. [Online]. Available: <https://www.brainvire.com/top-10-differences-between-wcf-and-web-services/>. [Accessed 3 May 2018].

FINAL GRADE

GENERAL COMMENTS

Instructor

74/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

GRADING FORM: SMU GENERIC\_GJ

LEAH CAHILL

74

RESEARCH-INFORMED LITERATURE



A good number of appropriate sources used.

4

## KNOWLEDGE AND UNDERSTANDING

- 

Some good knowledge and understanding evident. Not quite convinced by the use of a peer to peer application, but you've made it work. Not quite sure how useful it is as a web store, but it sure is interesting!
- 20

## ANALYSIS

- 

Good analysis - inclusion of encryption discussion in the report especially good.
- 17

## PRACTICAL APPLICATION AND DEP.

- 

A finished application. As mentioned above, a curious choice of paradigm, but ultimately meeting the requirements!
- 29

## SKILLS FOR PROFESSIONAL PRAC.

- 

A well-written, nicely presented report. Well done, Leah.
- 4