SABIC UK Petrochemicals/Teesside University


# Student Placement Report

**Stephen Pammenter**

Applications Developer

October 2012

# PREFACE

It is assumed that the reader of this report is from a technical background, and understands the terminology and syntax used throughout, without additional explanation required from the author. Code used in this document is represented using the `Courier New` font, and will be coloured in the same style as would be found in Microsoft Management Studio and Microsoft Visual Studio.

All examples of code used throughout this report is property of SABIC UK Petrochemicals[1]

[1] Registered Number: 03767075, Registered Office: The Wilton Centre, Redcar, Cleveland TS10 4RF.

# CONTENTS

# 1. INTRODUCTION

In June 2012 I began my placement for SABIC UK Petrochemicals, taking on the role of an Application's Developer in the Application's team of the Automation Department. It is to SABIC that I am contracted through Argent and Waugh, whom also provide Sabisu – a web based enterprise platform.

The applications team is here solely to provide IT solutions to SABIC UK; this comes in the form of both maintenance and development. It is with this team that I have been required to learn each system I am to maintain, and also how to develop to an enterprise standard. Much of the work I do in SABIC is with web based systems, which is reflected in the systems that I have worked on.

# 2. Main Systems Worked On

Many of the older systems have been developed using older technology, which has provided an insight into how it used be done and has presented a contrast in how modern development is performed. This in turn has provided both a richer knowledge of how some modern technology has been built up. The following systems are all new technologies; I have listed only these as they were the main systems I have been working on.

## 2.1 S18 Reporting

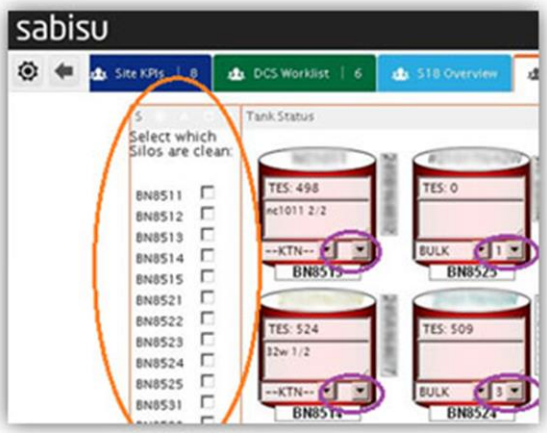


**Figure 1 - Tank Status Widget**

S18 is SABIC's LDPE plant at Wilton, dealing in both production and distribution. One of the systems used to track the details of this mass distribution is *S18 Reporting (Figure 1[2])*, which consists of front-end widgets in Sabisu, and a back end data-base that connects to various other plant systems around Wilton.

Sabisu is a third party web based enterprise platform built to seamlessly consolidate data. This system show's data through widgets on a screen - which are built by my team, using SQL, JavaScript, CSS and the Sabisu widget API. A large portion of my work orientates around Sabisu, as one of my job roles is a Widget editor. Widget editing is done through the widget editor section of the Sabisu platform, where the editor can select a component, drag it into the widget body, and set the components attributes.

Recently I have added various other additions to the *tank status* widget, which was built by my predecessor. This widget shows the state and logistic details of each tank in one easy to read format. Some additions I have added has been a *silo wash widget* (Orange ellipses), and an additional drop down lists (Lilac) shown in Figure 1. The wash widget when checked will change the name of a Silo to mark it as clean. This is done using the `writeback` attribute of the widget in the widget editor (*Figure* 2), which makes a call to a stored procedure used to invert a bit value I added to the table which contains the tank status details.

[2] Some values have been blurred to protect sensitive information.


**Figure 2 - Widget Editor Writeback Attribute in Use**

This proved difficult partially due to my inexperience of SQL at the time, and due to the limitations of Sabisu as a data writer. However the end result was successful and the end-users were very happy.

## 2.2 RAP

RAP is a vast third party system used to store details of everything from work permits and gas test results. The work I have done on RAP includes adjusting an existing widget to automatically adjust date points to be between the last seven days for when that view was selected in a Sabisu widget.

This was done through the widget editor, it was quite difficult to do as the editor can be difficult to debug, as it is heavily based on JavaScript. My first attempt at the problem began with:

```
$Eval($Eval($WidgetValue(RAP.Rap Report Graph.ddlReport)) == 3
? $Now(168):$WidgetValue(txtFrom))
```

With the Sabisu functions `$Eval` being an *IF* statement and `$Now` being the current time. This was poorly done, as it explicitly relied on a drop down list (txtFrom) having a value of 3. After some discussion with Tony, we arrived at a better conclusion; which involved adding an additional column to the database that this widget could read from. The end result incorporating some SQL:

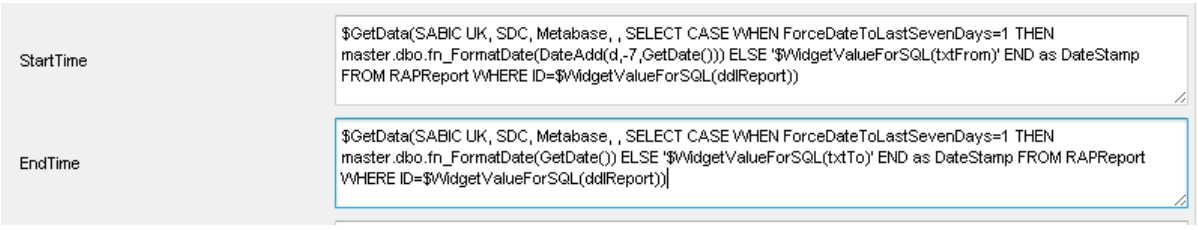


**Figure 3 - Widget Editor (Graph Start/End times)**

Here, an SQL *case when* takes the widget value form the drop down list of RAP views, uses that is in a select and decides whether to read a users selected date in another dropdown list, or substitute in our own date thus, forcing the graph to show correctly for the last seven days.

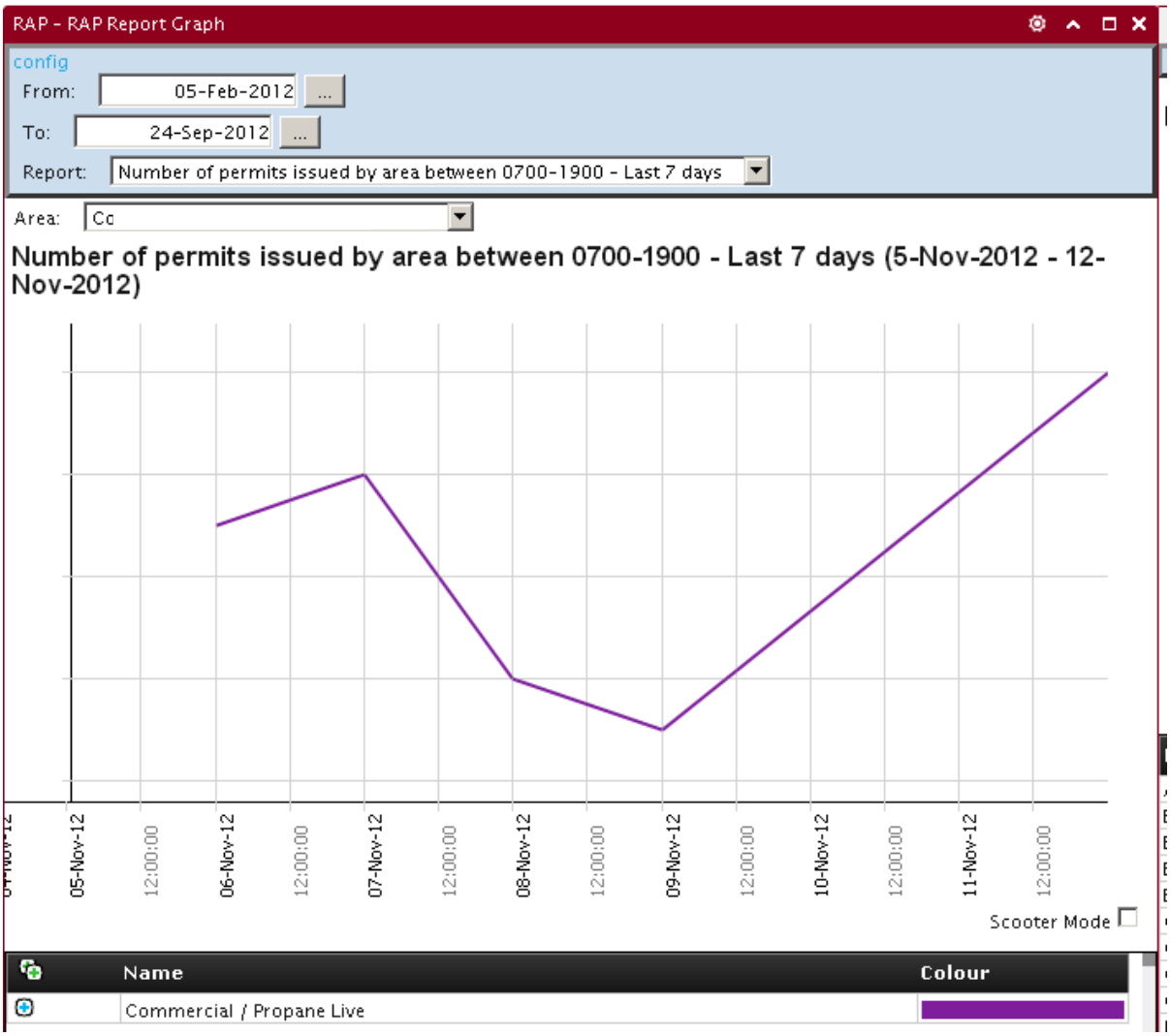


**Figure 4 - RAP Graph in Sabisu**

One of my bigger pieces of work was creating an agent to notify lab technicians when an outstanding gas test is due to be carried out. This was the first time I have built; Upon starting I, spoke to Tony for the best way to approach this. Together, we built an agent, and then he asked me to then duplicate the code we built from memory.

I enjoyed doing this as it was a great way to learn, and the end result was not that dissimilar from the original. The main difficulty from this work was ensuring that agent was running correctly with live data, without spamming a colleague's inbox with notifications. To combat this, I used server event logs to track the agent running. Additional code in the agent:

```
evnt.Source = "Site Plan Tracker Notification Agent";
evnt.Log = "Application";
WriteEventLog("Startup");
```

With a lot of communication between myself and the end-user, and the use of these logs, the end result was a clear e-mail, and a correctly working agent; allowing lab technicians to spend less time having to search for outstanding tests.

## 2.3 Site Plan Tracker

By far, this is the system I have spent the most work on. The system is used by the Site Leadership team to approve, define, track and review major site projects. It is built with a back-end database, server side C# code, an APSX front-end, and is then *thieved* into Sabisu using an iFrame. The development of this system was under full swing when I took over from the previous student. The development of which is done through an agile methodology, and guided by a Project Manager –again; with this placement another new experience for me.

I began with simple changes to the GUI, and soon I was working on a full iteration of the project to add full edit access to admin users. Over the weeks, I tweaked and added functionality to the server-side with buttons, and some additional validation using JavaScript. I then developed the database end stored procedures. I really enjoyed this 'full-frontal' programming, and this was really the start of getting hands on with a full system. *Figure 5* shows one of the screens in Site Plan. Here a delivery manager can add the project milestones. Originally this was a simple submission, but after the work I did, data in the screen is draft saved whenever a user goes to another screen -see ***Item 1***. And data can be re submitted in edit mode, all through one procedure.

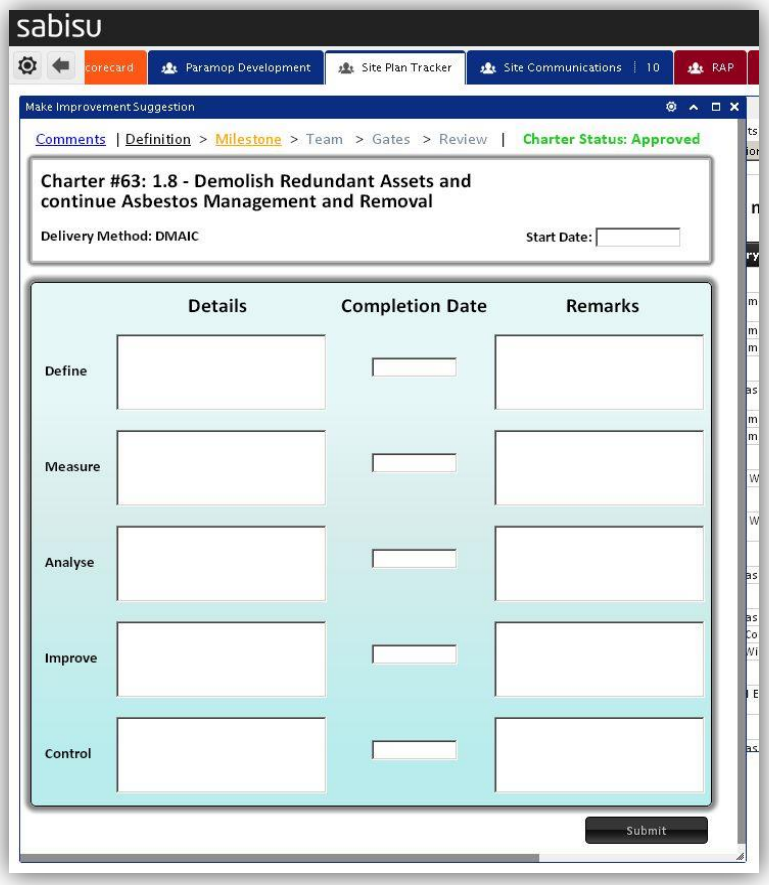


**Figure 5 - Milestone Screen in Sabisu iFrame**

The part of which I am most proud of, is the Improvement Blog, this tracks the stage changes of the improvement, and allows users to comment on the latest stage and provides a communication channel in which a delivery manager can

respond. This was very fun to build, as it wasn't very strongly defined in the system specification. Having a meeting with my manager Tony, we were able to throw ideas back and forth and I felt I had as much of a creative impact, as I did in building it. It also fantastically tied all the knowledge I had gained in the previous months into one neat package. The system incorporates a database, C# server-side backend, with an ASPX/HTML front end -whilst also making full use of an ASMX web services page.

I planned out how I wanted to do this on paper at first to get a feel for how the page would look, I then reproduced my final draft into 'bare bones' HTML, and applied CSS to get a real feel to how this page would look to an end user. It allowed me to make changes easily; before I built the back-end would dynamically produce this.

The back end took me two attempts, my first version dynamically created the page building up one long string of HTML. I felt this was not the best solution, so I requested a code review with Tony. From this, I then changed what was extremely hard to debug, and follow into a series of functions that created the page with generic HTML elements –see ***Item 2***.

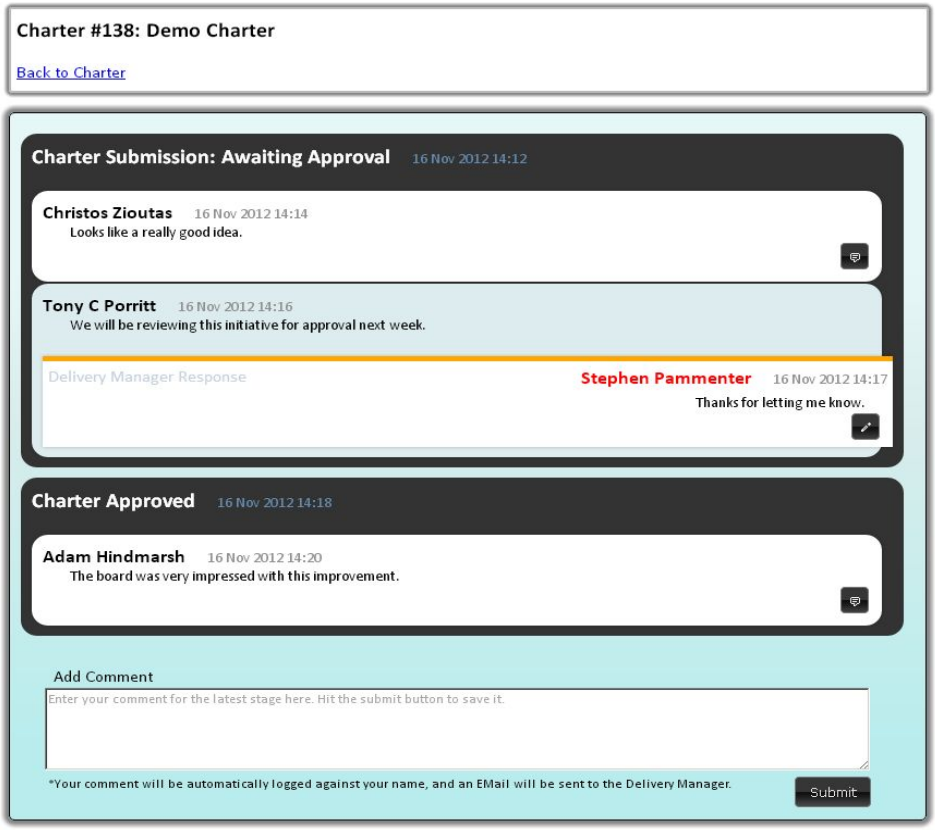


**Figure 6 - Site Plan Blog**

The end result was not only showing correctly, but was easy to debug and easy to make future changes to. The next problem I encountered was having dynamically created buttons send data to the server. Having the button connected to an *onClick* event in the server was all I knew at the time, which wasn't achievable with the current version. To combat this, I spoke to Tony who then trained me on the use of web service methods. Buttons now make an AJAX call to the web service page, which then processes the data using a post back -***Item 3***.

The completion of Site Plan has allowed me to meet all the expectations of my Employer; and has laid the foundations of what I expect to achieve, what is *expected* of me, a thorough

understanding of how various web technologies interlink with each other, and finally what to expect from a fully fledged agile project.

## 2.4 In Summary

Although many of the previous systems I have recorded and many of those not listed show a great account of the technical skills I have learnt in the time I have already spent on my placement, they do not reflect many of the soft skills I have also picked up. In contrast to my studies at University, and the relaxed tutorials spent with many of my peers; an office environment is much more professional. Communication is a vital part of my job, and without it work is much harder. I have ascertained much from troubleshooting over the phone with customers, and know the right questions to ask to solve problems much quicker.

Organisation, and management of my time and work has been paramount to ensuring projects are finished on time. And from the use of ticket manager (our work distribution system), calendar events and the direction of a project manager, I have learnt to organise time much better.

In three months I believe I have learnt more than I ever did in a year of university, my technical ability has come up leaps and bounds and my life skills have flourished. And I owe a great deal to my Manager and Supervisor Tony Porritt, who has given so much time to teach me many of these things.

I have thoroughly enjoyed the time already spent at SABIC, and greatly look forward to the challenges and lessons that lie ahead for the rest of the year.

# 3. Appendix

## Item 1. Site Plan Tracker - dbo.SaveMilestone

Here I have included the some of the additional code that I have added to the pre-existing stored procedure. The procedure checks for an existing charter to update, otherwise it inserts.

```
--Validate user has correct access to this charter, or is admin.
DECLARE @blnCorrectUser BIT;
SET @blnCorrectUser = (SELECT dbo.IsUserInRole(@CharterID, 'Delivery Manager'))
IF @blnCorrectUser = 0
	BEGIN
		SET @blnCorrectUser = (SELECT dbo.IsUserInRole(@CharterID, 'Team Leader'))
		IF @blnCorrectUser = 0
			SET @BlnCorrectUser = IS_ROLEMEMBER('Admin')
	END
IF @blnCorrectUser = 1
	BEGIN
		SET NOCOUNT ON
		SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
		--Update the data in the Milestone table, and give a date of submission
				UPDATE Milestone SET TargetCompletionDate =
				@CompletionDate,
				Comments = @Comments,
				Actions = @Actions,
				LastSaved = GETDATE()
				WHERE CharterID = @CharterID AND Name = @Name
		IF @@ROWCOUNT < 1
		BEGIN
		--No milestones where updated, so insert new...
		END
	END
ELSE
--Do not allow user access to the database...
```

## Item 2. Site Plan Tracker – Blog Page C# Server Back-end

The *CreateHTMLElement* function creates any generic HTML element required, and also treats specific arguments differently.

```
private HtmlGenericControl CreateHTMLElement(string ElementType, string ElementID, string
CssClass, string InnerText)
        {
            HtmlGenericControl htmlGenConNewElement = new
HtmlGenericControl(ElementType);

            string strSpanTag = "";
            string strInnerText;
            int intStringIndex;
            if (ElementID != "") htmlGenConNewElement.ID = ElementID;
            htmlGenConNewElement.Attributes.Add("class", CssClass);

    //We want to recognise the dvEdit as HTML, but not read everything else as HTML.
            intStringIndex = InnerText.IndexOf("<br/><div class="\dvEdit\">");
            if (intStringIndex > 0)
            {
                strInnerText = InnerText.Substring(0, intStringIndex);
                strSpanTag = InnerText.Substring(intStringIndex);
            }
            else strInnerText = InnerText;

            //We want to display the deleted span tag as HTML.
            if (strInnerText == "<span style=\"color: #66747B;\">[User Deleted
Comment]</span>")
              {
                    htmlGenConNewElement.InnerHtml = strInnerText;
              }
            else if (strInnerText != "")
              {
                    htmlGenConNewElement.InnerText = strInnerText;
              }

            if
              {
                    (strSpanTag != "") htmlGenConNewElement.InnerHtml += strSpanTag;
              }

            return htmlGenConNewElement;
        }
```

The remainder of the blog back end is simply iterations that build the structure of the web page, through *skeleton* generic variables. By doing it this way, the program saves a large amount of memory the much iteration it performs. Here is an example for a new thread:

```
//New Thread
if (drComment["AuthorID"] is DBNull)
{
       htmlDivThread = new HtmlGenericControl("div");
       htmlDivThread.Attributes.Add("class", "dvThread");
       htmlDivThreadHeader = new HtmlGenericControl("div");
       htmlDivThreadHeader.Attributes.Add("class", "dvThreadHeader");
       htmlSpanThreadName = new HtmlGenericControl("span");

//And so on...
```

## Item 3. Site Plan Tracker – Blog Page Web Service And JavaScript Call

Here I have shown the *onClick* event called when a user adds a new comment to the page. First the JavaScript is called:

```
function addComment() {
    var comment = $('#txtAddComment').val();
    var cid = $('#hdnCID').val();
    //Check to see if a comment exists.
    if (comment.length == 0) {
        alert("You need to enter a comment.");
    }
   //Use AJAX function to pass the data to the AddComment web method.
    else
        $.ajax(
            {
                url: 'https://' + GetDomain() + '/BlogWebService.asmx/AddComment',
                data: JSON.stringify({ CID: cid, Comment: comment }),
                type: 'POST',
                dataType: 'json',
                async: false,
                contentType: 'application/json; charset=utf-8"',
                success: function () {
                },

                error: function () {
                     alert('Error: There was a problem posting that comment.');
                }
            });
}
```

Using an AJAX call, I can then pass the details that I require from the front end to the Web Service. BuisnessReports is an in-house library we use for many of our applications. By placing various strings in the settings of the application, it allows developers to make changes on the server without having to through the five stage process of re-deploying with SVN.

```
[WebMethod]
        public void AddComment(int CID, string Comment)
        {
            string strSQL;
            string strEmailHTML;
            DataRow drOne;
            BusinessReports.Report bizReport = new BusinessReports.Report();
            bizReport.Connect(BusinessReports.Report.ConnectionType.SQLServer,
SabisuSitePlanTracker.Properties.Settings.Default.Server,
SabisuSitePlanTracker.Properties.Settings.Default.Database);
             //Use .replace() to SQL injection proof entered data.
            strSQL = "EXEC  [dbo].[AddComment] @CharterID =" + CID.ToString() + ",@CommentID = NULL,
@Author = '" + User.Identity.Name.Replace("'", "''")  + "', @Comment ='" + Comment.Replace("'", "''")
+ "', @Response = NULL";
            bizReport.ExecSQL(strSQL);

            drOne = bizReport.GetDataRow("SELECT TOP 1
[CharterID],[CharterName],[AuthorFullName],[Comment],[DeliveryManagerFullName],[Response] FROM
[SitePlanTracker].[dbo].[CommentWithManagerFullName] WHERE CharterID =" + CID + " ORDER BY ID DESC");
       //Send email to the delivery manager to notify them of a new comment on their initiative.
            strEmailHTML = Properties.Settings.Default.NewCommentEMail;
            strEmailHTML = strEmailHTML.Replace("$Author$", drOne["AuthorFullName"].ToString());
            strEmailHTML = strEmailHTML.Replace("$CID$", CID.ToString());
            strEmailHTML = strEmailHTML.Replace("$CharterName$", drOne["CharterName"].ToString());

            bizReport.SendEMail(drOne["DeliveryManagerFullName"].ToString().Replace(" ", ".") +
"@SABIC-Europe.com", "Site Plan Notification: Comment made on your Improvement", strEmailHTML);

            bizReport.Disconnect();
        }
```