CASE STUDY

Aspose.PDF for .Net Case Study

Mainpac Solutions Pty Ltd

Using Aspose.PDF for .Net to generate an editable PDF document

Angelo Vargheese, Development Manager, 25 May 2021

About Mainpac Solutions

Established in 1984, Mainpac is an Australian software vendor which is now part of the Vela Software Group, a wholly owned subsidiary of Constellation Software, Inc. (TSX:CSU), an international provider of software and services to multiple industries across public and private sectors.

Mainpac provides Enterprise Asset Management (EAM) solutions across Australia and within 23 international markets to asset-intensive industries including manufacturing, mining and resources, ports and terminals, utilities, government, and facilities management. Mainpac's EAM on-premise and on-demand software solutions are complemented by consulting, implementation, training, and support services.

Mainpac EAM allows planning and control of all the activities required to maintain business assets throughout each individual asset's lifecycle. These activities include, but are not limited to, the scheduling of planned maintenance, optimization of asset productivity, documentation of information in relation to breakdown and malfunctions, stock control, inventory, purchasing information and many more features.

Problem

In Mainpac EAM users can create and assign Work Orders (Jobs) to field technicians. In most situations a field technician will have a user account to the Mainpac EAM application and can update the Work Order details using the application when they finish the job.

Where a customer has many field technicians, across multiple organisations assigning and defining each of these resources can become an administrative burden, therefore an alternative approach to paper-based work orders, is the use of the application to send out emails to field technicians with an attached PDF.

The attached PDF is generated by a Microsoft SQL Server Reporting Services (SSRS) report.

The PDF has the details of the job and sections that must be filled out by the technician once they complete the job, including details of the work that they have done, the parts they have used, the condition of the asset and any additional costs involved.

The field technician would typically print out the PDF, fill out the details, then scan and email the document back to the company. The company would then have an employee manually renter the details back into the application.

This process was causing inaccurate data to be entered into the application and also added the expense of employing a person for data entry to the customer.

How important was it to solve these problems?

This problem has been raised as a high priority by our customers that mainly use external contractors to complete their field based activities.

How many users were involved, how many customers affected?

Customers typically dealing with 100’s of contractors and generating between 200 and 400 field work orders per month.

What were your objectives?

- Generate an editable PDF for a Work Order from the application.
- Ensure data is entered correctly in the editable PDF, by providing drop down fields, validating data types for dates, time and numeric fields and confirming that all required fields are entered.
- Read the completed PDF and update the Work Order details in the application automatically.

Solution

After evaluating a few other PDF generation components, we found that the Aspose.PDF .NET Component was the easiest to use to generate editable PDFs.

The Aspose.PDF .NET component provided the ability to flag the fields as required and to also attach JavaScript functions to validate each field individually or validate against multiple related fields.

Experience

Finding a solution:

Aspose.PDF was among a few products that we found using a google search for .net components to generate PDF Forms.

We were able to use the evaluation version of Apose.PDF and the sample code provided on their GitHub page to confirm that it had all the features that we required.

The other products we evaluated were able to generate PDF Forms but did not have information on how to implement validation on the fields, whereas we able to find several posts on the Aspose forum on how to add validation.

Implementation:

We were able to continue to use SSRS to create an initial PDF with just the Work Order details.

Then using APOSE.PDF we added a new page to the initial PDF and added the editable fields and labels to this new page.

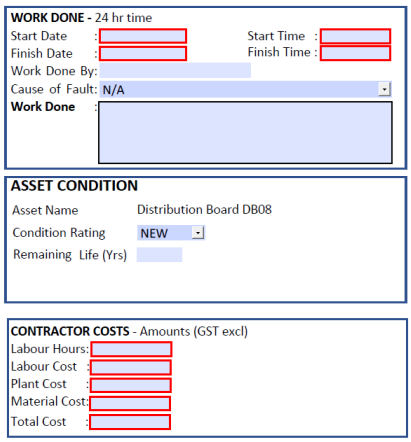


Figure 1: Work Order Feedback Page

We were able to add data type (date, time, number formats) validation to the fields using the OnValidate action and the standard adobe JavaScript functions AFDate, AFTime etc,.

```
PdfAction action = new JavascriptAction("AFDate_FormatEx(\"dd/mm/yyyy\");");
textBoxField.Actions.OnValidate = action;
textBoxField.Actions.OnModifyCharacter = new JavascriptAction("AFDate_KeystrokeEx(\"dd/mm/yyyy\");");
```

Figure 2: Adding Date Format Validation

To validate between two fields, e.g End Date must be after Start Date, we able to write our own JavaScript and attach to the LostFocus action.

```
var onLostFocusScript = @"var one = this.getField('StartDate');
                          var two = this.getField('EndDate');
                          var date1=  util.scand('dd/mm/yyyy', one.value);
                          var date2=  util.scand('dd/mm/yyyy', two.value);
                          if (date1>date2) app.alert('Start date cannot be after end date.'); two.value='';";
endDate.Actions.OnLostFocus = new JavascriptAction(onLostFocusScript);

document.Form.Add(endDate, 1);
```

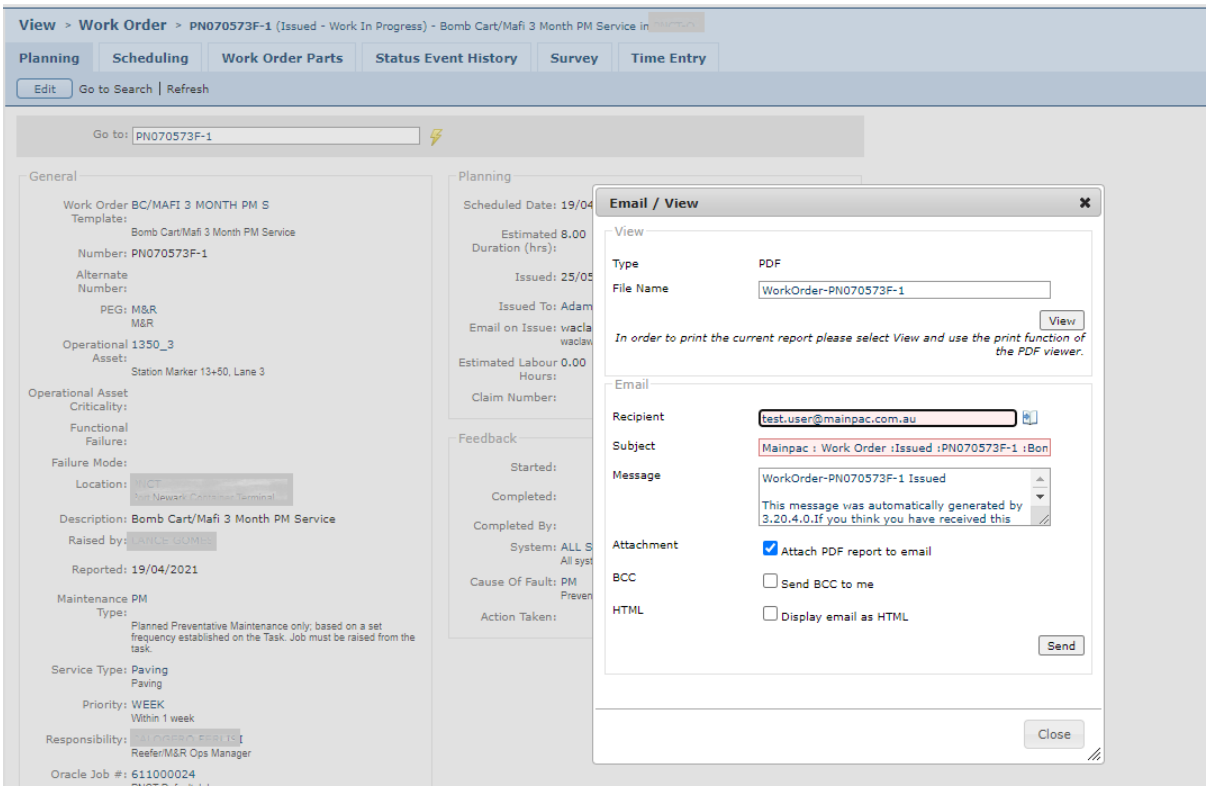
Figure 3: Add custom validation.

We added the Work Order ID as a hidden field so that we had a reference id when we read the completed PDF and needed to update the Work Order in the application.

```
TextBoxField hiddenFieldWorkOrderId = new Aspose.Pdf.Forms.TextBoxField(doc, new Rectangle(0, 0, 0, 0));
hiddenFieldWorkOrderId.PartialName = "WorkOrderId";
hiddenFieldWorkOrderId.Value = workOrderId.ToString();
hiddenFieldWorkOrderId.Flags = AnnotationFlags.Hidden;
doc.Form.Add(hiddenFieldWorkOrderId, 1);
```

Figure 4:Add hidden field to document.

Generating the PDF in the server-side code enabled us to continue using our existing UI to send the PDF through email.



Screen 5 : Send Email with Editable PDF

Once the completed PDFs were sent back, they were placed in a folder, where we used Apose.Pdf in background job to open the file and simply iterated through the Forms collection on Aspose.Pdf.Document class to update the Work Order in the application

It took roughly 3-4 days to implement this functionality. The development team has found it relatively easy to use Aspose.PDF. They have been able to just use the Aspose documentation web page and forum to code the required functions.

Outcome:

Using Aspose.PDF we were able to easily add the editable pdf functionality into our product. Most of our customers have found this feature a useful addition to the product.

Next Steps

We will be looking use Apose.Pdf to generate editable PDFs in other areas of our application.

Summary

Overall, we have been very happy with Apose.PDF .Net and can only recommend component to any other .Net developers. The product was easy to work with and our time to market greatly reduced, delivering an overall better customer experience.