

Create and load PDF Forms in Universal Applications using Windows Store and Windows Phone PDF library

Written by Apitron Documentation Team

Introduction

Universal Windows Store application project provides a good way to share code and develop Windows Phone and Windows Store version of the application at the same time. In case of our library it's even more fun because API remains the same and you can use the same PDF .NET library for both projects.

PDF files can contain forms consisting of fields where the information can be stored for later use and processing. If you'd like to read more about PDF forms and fields, official PDF specification is an invaluable resource. This topic is also covered in our pdf cookbook "Aptron PDF Kit in action" (available for free by the following [link](#)), see sections 3.7 "*Interactive Forms*" and section 4.4 "*Content Element And Controls*". There are examples on how to create, edit or read data stored in PDF forms, written in C# with detailed explanation.

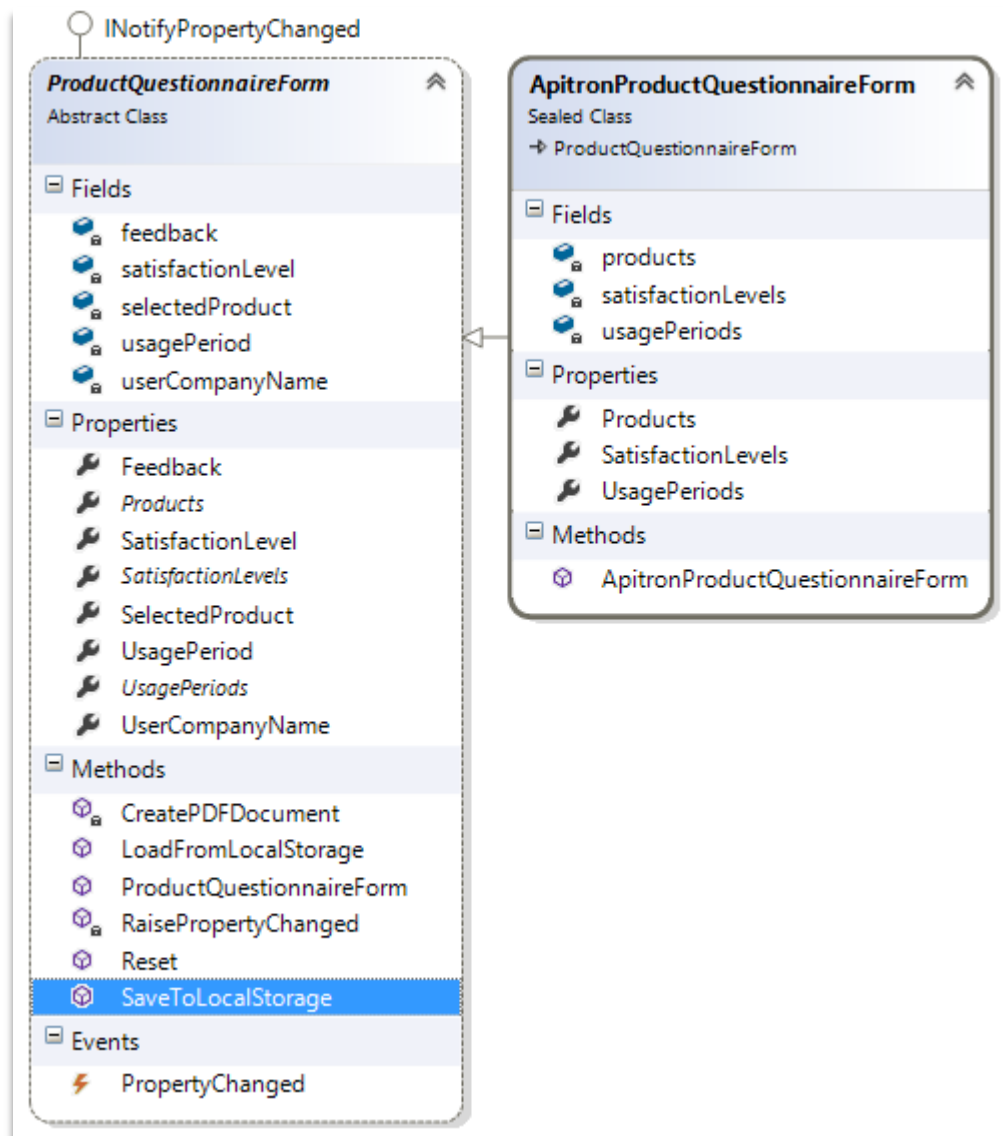
A good example demonstrating how to work with PDF forms is a questionnaire app gathering some data from the user and storing it in PDF document as a form. So we decided to go through and prepared sample universal app using our Aptron PDF Kit library that generates, saves and loads product questionnaire data.

Solution overview

To keep this sample as simple as possible we used single XAML page representing form layout and providing the data bound view for questionnaire data.

Data Model

Data model is represented by base class called `ProductQuestionnaireForm` and by its specific implementation `ApitronProductQuestionnaireForm`. See the diagram below.



Pic. 1 Class digram, Apitron questionnaire sample

Page controls are bound to properties of `ApitronProductQuestionnaireForm` instance, so we don't have to write any code to handle form updates if user changes their content.

Windows Phone App

The view is represented by XAML app page, see picture below showing desktop and phone versions. They are almost identical and use same markup except some minor changes needed to fit content on the phone screen.

Apitron PDF Kit for .NET sample, saving and loading questionnaire as PDF form

Product:

Apitron PDF Kit for .NET

Usage period:

One year or more

Satisfaction level:

High

Your company name (optional):

Apitron

Feedback (optional):

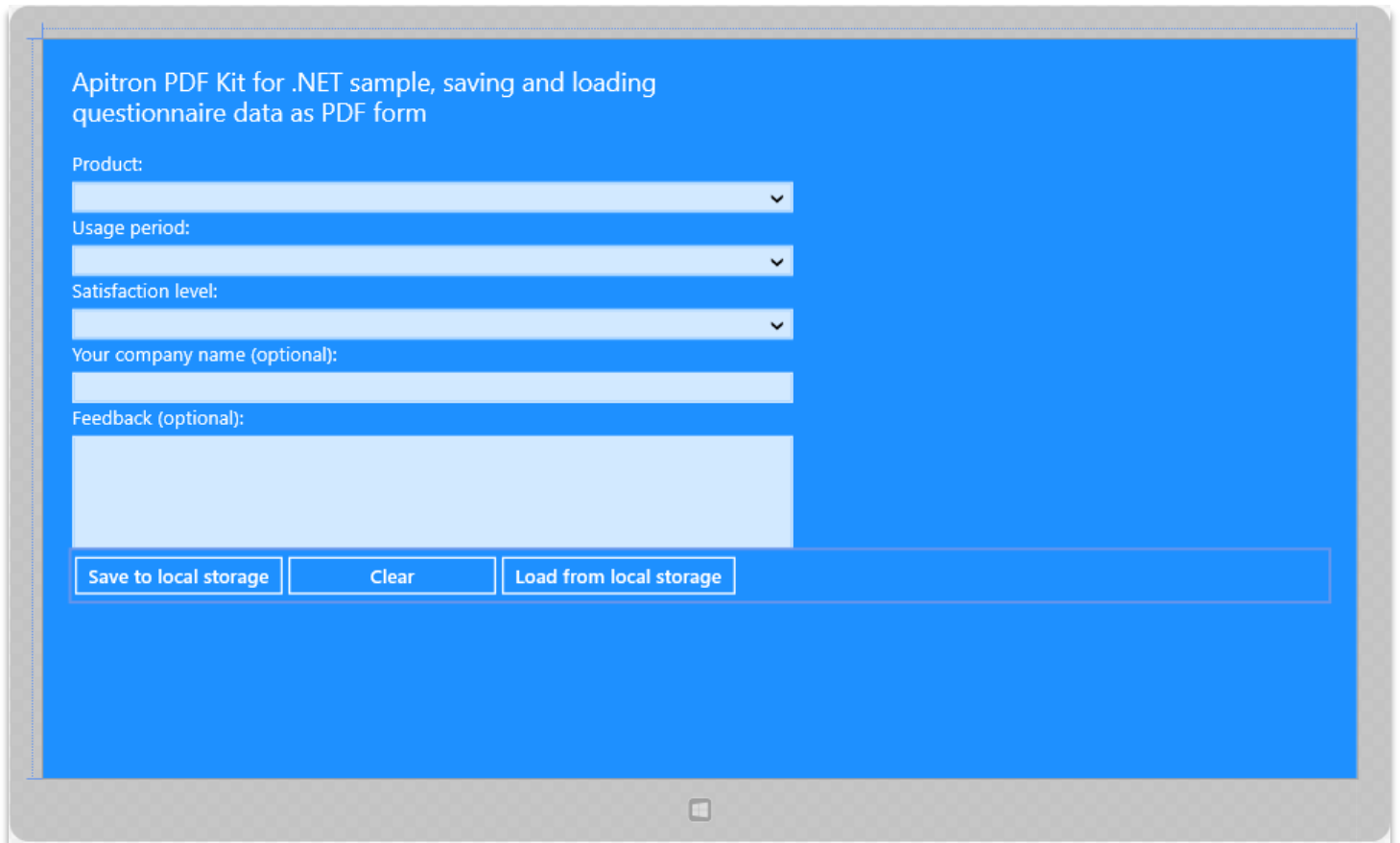
I'll recommend it to my co-workers

Save Clear Load

Pic. 2 Questionnaire App, Windows Phone version

Windows Store App

Form layout for Windows Store app is shown here, below is a screenshot from Visual Studio XAML designer.



Apitron PDF Kit for .NET sample, saving and loading questionnaire data as PDF form

Product:

Usage period:

Satisfaction level:

Your company name (optional):

Feedback (optional):

Pic. 3 Questionnaire app, Windows Store version

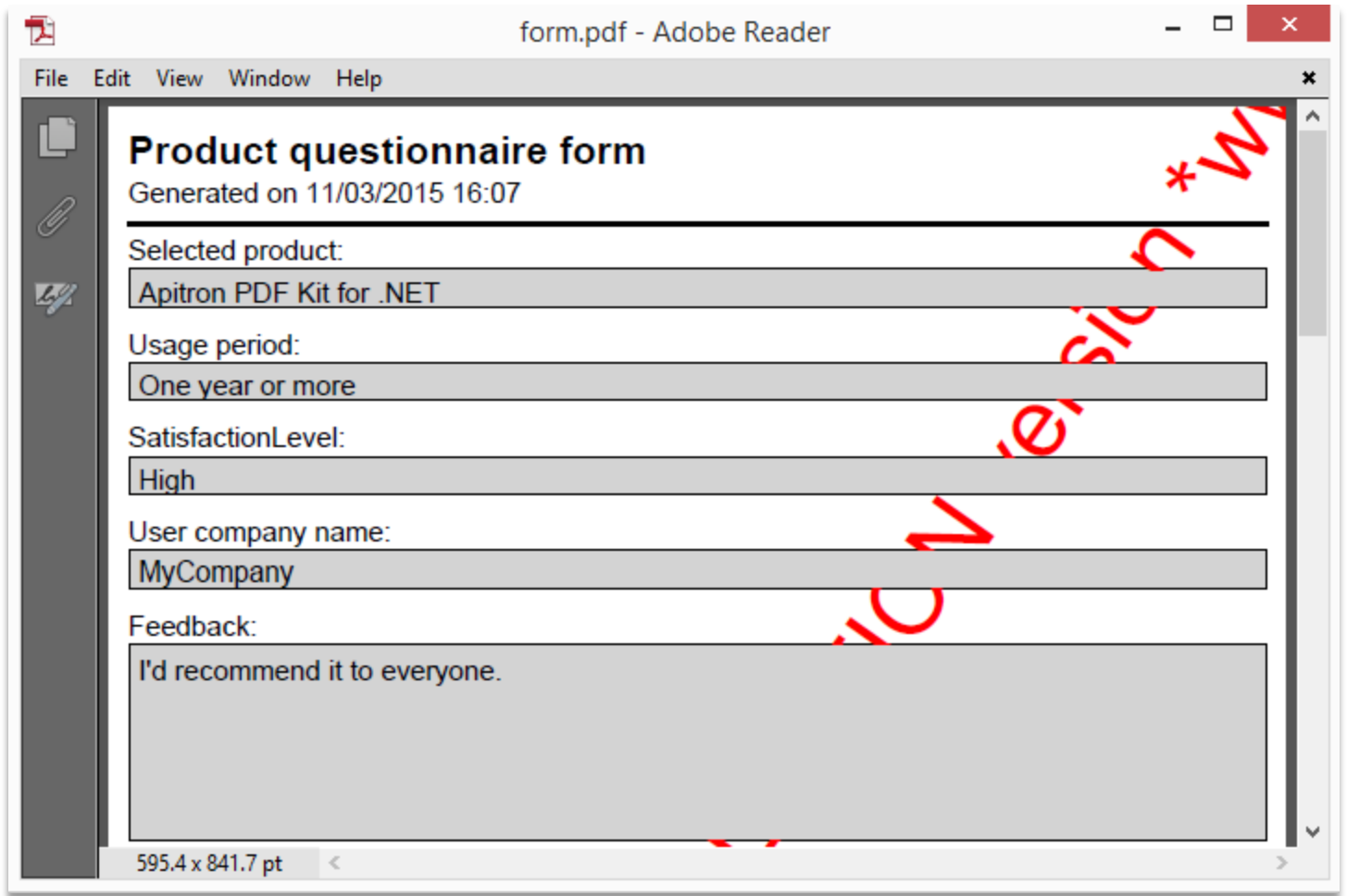
Workflow

Using this app the user can fill the questionnaire form provided, in our case it's a test form collecting opinions about Apitron products. After that, he can save entered data to local app storage as PDF form or load an existing form from it.

For simplicity, data is being saved as file "form.pdf" located in local app folder, which can be opened by using the following "known" folder: `ApplicationData.Current.LocalFolder`.

Output

Generated questionnaire pdf form is shown on the image below. Both apps generate identical PDF files.



The image shows a screenshot of the Adobe Reader application window titled "form.pdf - Adobe Reader". The window displays a PDF form titled "Product questionnaire form" with a timestamp "Generated on 11/03/2015 16:07". The form contains the following fields and their values:

- Selected product:** Apitron PDF Kit for .NET
- Usage period:** One year or more
- SatisfactionLevel:** High
- User company name:** MyCompany
- Feedback:** I'd recommend it to everyone.

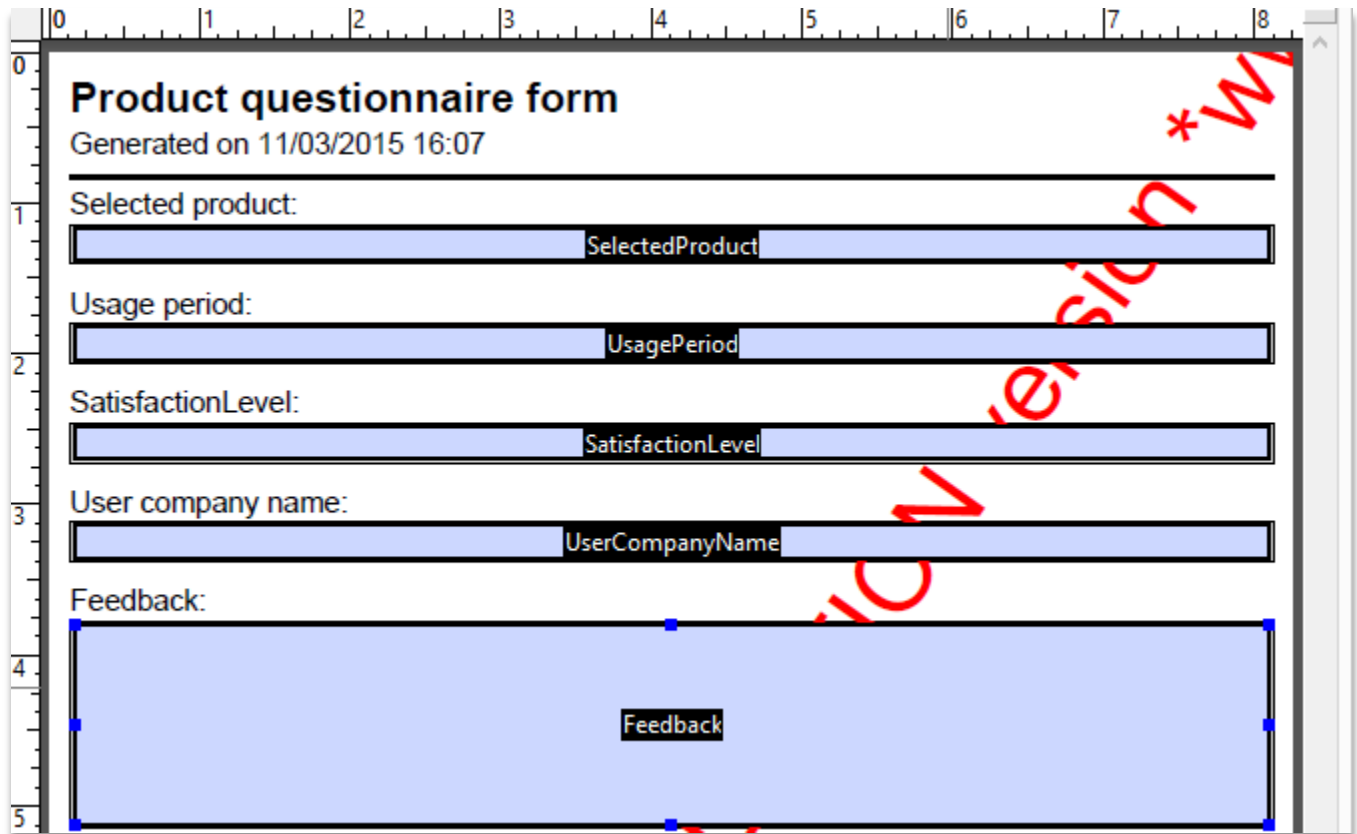
The form is displayed in a standard Adobe Reader interface with a menu bar (File, Edit, View, Window, Help) and a sidebar on the left. A large red watermark "WATERMARK *W" is visible diagonally across the form content.

Pic. 4 Generated questionnaire pdf form

This file can be found by the following path:

C:\Users\[user name]\AppData\Local\Packages\[deployed package guid]\LocalState\form.pdf

Same form but opened in design mode is shown below. You can see field names assigned to each widget annotation present on page. Widget annotations are special kind of controls used to define visual representation of form fields data. In our case all controls created for this PDF form are made read-only, however it's possible to make form data editable after generation if needed.



Pic. 5 Generated questionnaire pdf form opened in design mode

The code

Complete app sample can be found in Apitron PDF Kit download package available on our website, check out this [link](#).

Here we'd like to highlight PDF specific code that is responsible for form generation. It can be found in ProductQuestionnaireForm class and is provided below.

PDF Form generation and saving

```
/// <summary>
/// Creates FlowDocument that represents PDF form.
/// </summary>
private FlowDocument CreatePDFDocument()
{
    FlowDocument document = new FlowDocument(){Margin = new Apitron.PDF.Kit.Styles.Thickness(10)};

    // add form fields and set text values based on form data
    document.Fields.Add(new TextField("SelectedProduct", SelectedProduct){IsReadOnly = true});
    document.Fields.Add(new TextField("UsagePeriod", UsagePeriod) { IsReadOnly = true });
    document.Fields.Add(new TextField("SatisfactionLevel", SatisfactionLevel) { IsReadOnly = true });
    document.Fields.Add(new TextField("UserCompanyName", UserCompanyName) { IsReadOnly = true });
    document.Fields.Add(new TextField("Feedback", Feedback) { IsReadOnly = true, IsMultiline = true});

    // register styles defining controls appearance
    document.StyleManager.RegisterStyle(".formHeader",new Apitron.PDF.Kit.Styles.Style()
{Display = Display.Block, Font = new Font(StandardFonts.HelveticaBold, 20)});
    document.StyleManager.RegisterStyle("TextBlock, TextBox", new Apitron.PDF.Kit.Styles.Style()
{ Font = new Font(StandardFonts.Helvetica, 14)});
    document.StyleManager.RegisterStyle("TextBlock + TextBox", new Apitron.PDF.Kit.Styles.Style()
{ BorderColor = RgbColors.Black, Border = new Border(1), Height = 20, Background =
RgbColors.LightGray});
    document.StyleManager.RegisterStyle("Br", new Apitron.PDF.Kit.Styles.Style() { Height = 10});

    // add document content elements
    document.Add(new TextBlock("Product questionnaire form"){Class = "formHeader"});
    document.Add(new TextBlock("Generated on " + DateTime.Now.ToString("dd/MM/yyyy HH:mm")));
    document.Add(new Hr(){Height = 2, Margin = new Thickness(0,5,0,5)});
    document.Add(new TextBlock("Selected product:"));
    document.Add(new TextBox("SelectedProduct"));
    document.Add(new Br());
    document.Add(new TextBlock("Usage period:"));
    document.Add(new TextBox("UsagePeriod"));
    document.Add(new Br());
    document.Add(new TextBlock("SatisfactionLevel:"));
    document.Add(new TextBox("SatisfactionLevel"));
    document.Add(new Br());
    document.Add(new TextBlock("User company name:"));
    document.Add(new TextBox("UserCompanyName"));
    document.Add(new Br());
    document.Add(new TextBlock("Feedback:"));
    document.Add(new TextBox("Feedback"){Height = 100});

    return document;
}
```



```

/// <summary>
/// Saves current form data to local storage.
/// </summary>
public async Task<bool> SaveToLocalStorage(string fileName)
{
    try
    {
        FlowDocument pdfDocument = CreatePDFDocument();

        StorageFile file = await
ApplicationData.Current.LocalFolder.CreateFileAsync(fileName, CreationCollisionOption.ReplaceExisting);

        using (Stream outputStream = await file.OpenStreamForWriteAsync())
        {
            pdfDocument.Write(outputStream, new ResourceManager(), new PageBoundary(Boundaries.A4));
        }
    }
    catch (Exception e)
    {
        Debug.WriteLine(e.Message);
        return false;
    }

    return true;
}

```

Loading stored PDF form data

```

/// <summary>
/// Load form data from PDF file using given filename.
/// </summary>
public static async Task<ProductQuestionnaireForm> LoadFromLocalStorage(string fileName)
{
    try
    {
        StorageFile file = await ApplicationData.Current.LocalFolder.GetFileAsync(fileName);

        using (Stream inputStream = await file.OpenStreamForReadAsync())
        {
            // use fixed document for reading form data
            FixedDocument pdfDocument = new FixedDocument(inputStream);

            // create and initialize form object instance
            ProductQuestionnaireForm form = new ApitronProductQuestionnaireForm();

            form.SelectedProduct = ((TextField)pdfDocument.AcroForm["SelectedProduct"]).Text;
            form.UsagePeriod = ((TextField)pdfDocument.AcroForm["UsagePeriod"]).Text;
            form.SatisfactionLevel = ((TextField)pdfDocument.AcroForm["SatisfactionLevel"]).Text;
            form.UserCompanyName = ((TextField)pdfDocument.AcroForm["UserCompanyName"]).Text;
            form.Feedback = ((TextField)pdfDocument.AcroForm["Feedback"]).Text;

            return form;
        }
    }
    catch (Exception e)
    {
        Debug.WriteLine(e.Message);
    }

    return null;
}

```

Conclusion

This approach can be used to create PDF forms in Windows Forms, WPF, Xamarin (Android and iOS), console, web or cloud applications. As you can see from samples above it takes *less than 30 lines of code* to produce a basic form having all necessary content set.

But Apitron PDF Kit library is not limited to just PDF forms, it provides full support for PDF features defined in official specification and can be used to create apps targeting all of the modern mobile, desktop or web platforms. Free evaluation is available for all our products.

We'd be happy to get any feedback from you, feel free to contact our support with any questions you may have.