

Create PDF Forms in iOS applications using Xamarin PDF library by Apitron

Written by Apitron Documentation Team

Introduction

In addition to Xamarin Android application that was recently highlighted in [one of our articles](#), we created a similar sample for iOS. It uses Apitron PDF Kit for pdf form generation and its PDF related code and data model are the same as used for Android app. The UI part is, of course, different but it could be generalized using Xamarin.Forms if needed.

The complete sample can be found in `Samples\Xamarin.iOS` folder inside the download package available on our [website](#). It's called `CreateQuestionnaireFormSample`.

Note: this application has been tested using Apitron PDF Kit for .NET(Xamarin build) version 1.0.6 and iPhone iOS simulator device with iOS 7.1 and iOS 8.2 installed.

Solution overview

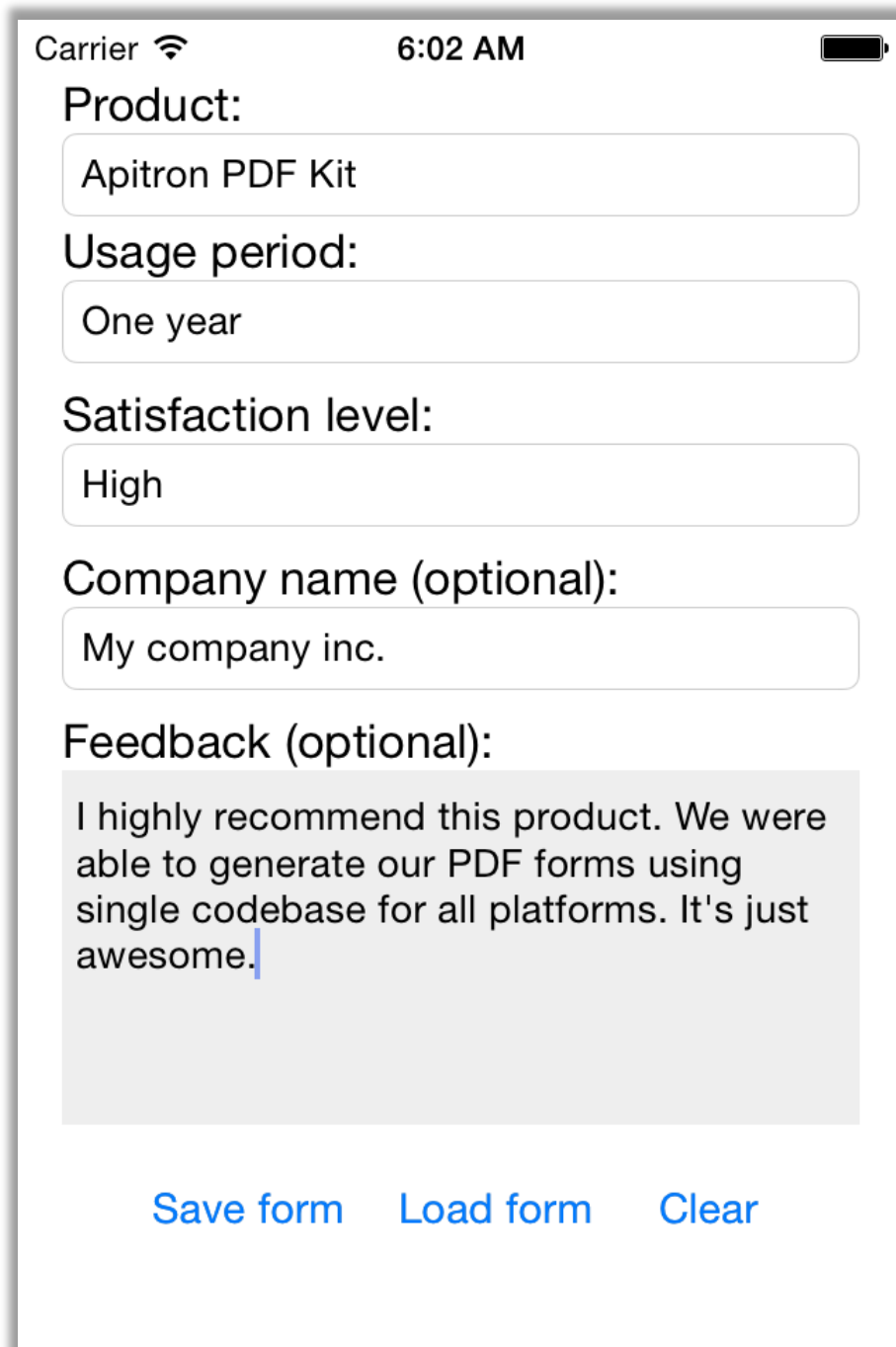
Our single-view storyboard-based application starts with the launchscreen defined in launchscreen.xib and then transits to the main storyboard.



Pic. 1 PDF form generation, iOS app demo, launch screen

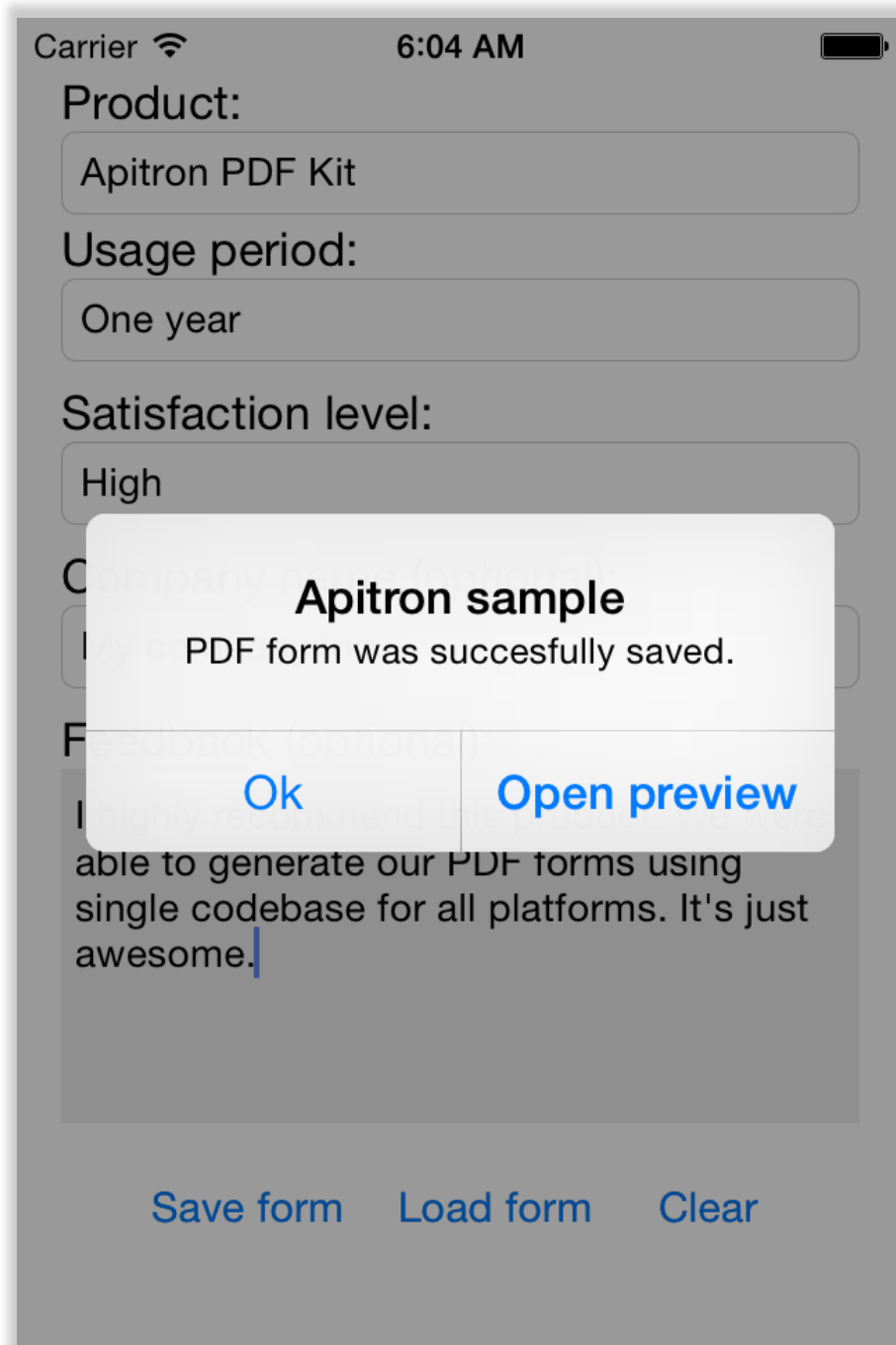
The demo iOS application UI is implemented using the single storyboard containing several UITextField objects linked to corresponding outlets. These fields are being used for capturing the data and creation of the PDF form as well as for displaying loaded form data. By touching “Clear” one may reset the form.

See the device screenshot below:



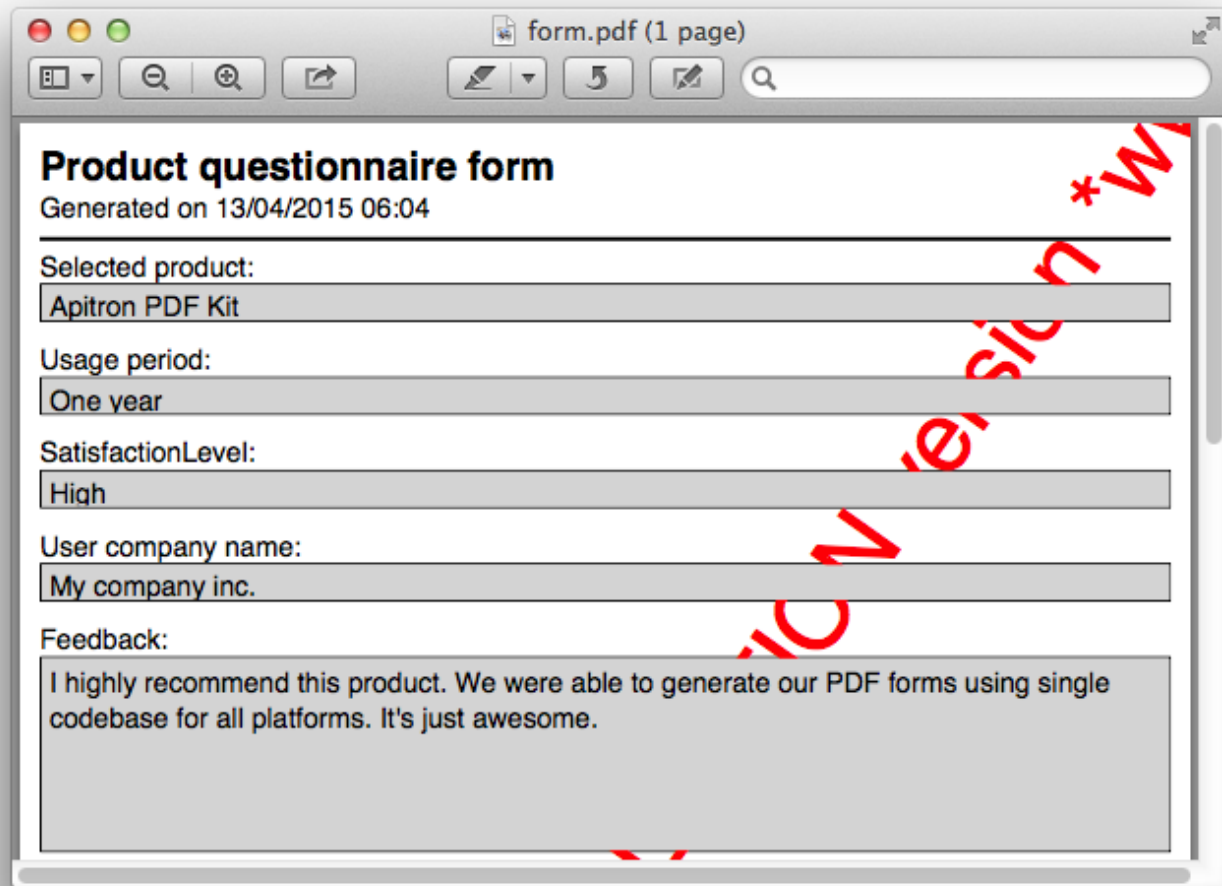
Pic. 2 PDF form generation, iOS app demo, main screen

When the user clicks “Save form” it starts pdf form generation. UIAlertView notifies whether the operation was completed successfully and offers to open the default preview if needed. For some reason iOS QLPreviewController on simulator doesn’t correctly show the created PDF form (it displays empty fields) while the regular OS X preview app does. Adobe Acrobat Reader opens this file without any problems. See the device screenshot below:



Pic. 3 PDF form generation, iOS app demo, generation completed

Resulting PDF file downloaded from the iPhone simulator device is shown below:



Product questionnaire form
Generated on 13/04/2015 06:04

Selected product:
Apitron PDF Kit

Usage period:
One year

SatisfactionLevel:
High

User company name:
My company inc.

Feedback:
I highly recommend this product. We were able to generate our PDF forms using single codebase for all platforms. It's just awesome.

Pic. 4 Resulting PDF form generated by CreateQuestionnaireSampleForm iOS application

Form fields are made read-only so they can't be changed in reader app after generation. These fields store the information entered by user and can be used to retrieve the information back. This is how "Load form" works. Resulting file is being written to directory returned by the `Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)` call.

The code

Complete Xamarin iOS app code can be found in download package while here only the PDF generation part is being shown. It returns `FlowDocument` object which can later be saved.

```
// Creates FlowDocument that represents PDF form.
private FlowDocument CreatePDFDocument()
{
    // create document with margin
    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)});
    // set style for textblock and textbox content elements using type selectors
    document.StyleManager.RegisterStyle("TextBlock, TextBox", new Apitron.PDF.Kit.Styles.Style() {
Font = new Font(StandardFonts.Helvetica, 14)});
    // set style for textbox followed by a textblock using adjacent element selector
    document.StyleManager.RegisterStyle("TextBlock + TextBox", new Apitron.PDF.Kit.Styles.Style()
{ BorderColor=RgbColors.Black, Border=new Border(1), Height = 20, Background = RgbColors.LightGray});
    // set br style using type selector
    document.StyleManager.RegisterStyle("Br", new Apitron.PDF.Kit.Styles.Style() { Height = 10});

    // add form header
    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)});
    // add product info content
    document.Add(new TextBlock("Selected product:"));
    document.Add(new TextBox("SelectedProduct"));
    document.Add(new Br());
    // add usage info content
    document.Add(new TextBlock("Usage period:"));
    document.Add(new TextBox("UsagePeriod"));
    document.Add(new Br());
    // add satisfaction level content
    document.Add(new TextBlock("SatisfactionLevel:"));
    document.Add(new TextBox("SatisfactionLevel"));
    document.Add(new Br());
    // add company name content
    document.Add(new TextBlock("User company name:"));
    document.Add(new TextBox("UserCompanyName"));
    document.Add(new Br());
    // add feedback content
    document.Add(new TextBlock("Feedback:"));
    document.Add(new TextBox("Feedback"){Height = 100});
    return document;
}
```

Load pdf form back from pdf file:

```
/// <summary>
/// Load form data from PDF file using given filename.
/// </summary>
public static ProductQuestionnaireForm Load(string filePath)
{
    try
    {
        using (Stream inputStream = File.OpenRead(filePath))
        {
            // 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

If you need an easy to maintain and cost-effective PDF processing solution for targeting many platforms at once, Apitron PDF Kit is a good choice. It allows you to create applications and services for all modern desktop, cloud and mobile platforms. You can download the evaluation version with lots of C# samples demonstrating the API from our website ([link](#)).

We've also written a [free book](#) for you describing the API and containing ready to use code snippets. It covers most parts of the PDF specification with detailed explanations and provides real life usage examples. [Contact us](#) if you are interested or have any feedback.