Digitally sign PDF documents using InkManager in Windows Store applications

Written by Apitron Documentation Team

Introduction

Signing PDF from scratch is a complex task and it may take some time to implement it properly. We made it much simpler with our Apitron PDF Kit for .NET library, whenever you need to add a digital signature to PDF document you can use our library and get it done in a minute. To help you on the way, we've created a complete Windows Store Application C# sample showing how to process ink drawings captured by `InkManager` defined in `Windows.UI.Input.Inking` namespace.

The complete sample can be found in *Samples\Microsoft Windows Store 8.1* folder inside the download package available on our website. It's called *CaptureSignatureAsInkDrawing.*

Solution overview

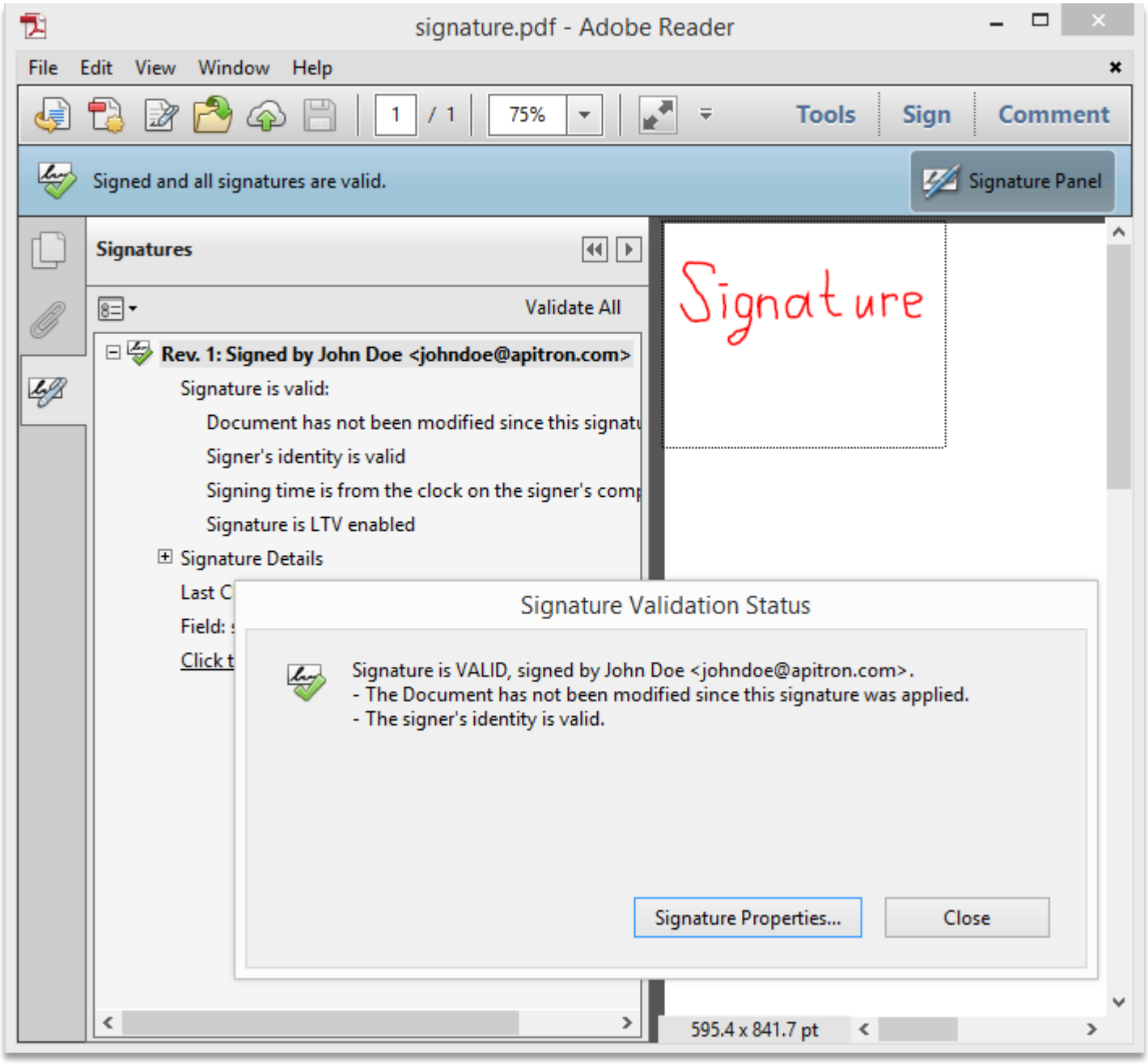
Demo app layout is plain, a white panel and two buttons, one to save the resulting signature and create new PDF file and one for clearing the signature panel.

See the screenshot below:

Pic. 1 Capture signature using Ink Manager, sample windows store app

One can draw any signature and click or touch (if you run it on Windows Surface or similar tablet) *Save to PDF* button. When it happens, captured signature data becomes processed and converted to PDF drawing commands. Resetting the pane is as simple as clicking *Clear*.

Resulting PDF file contains a signature field with its visual representation placed on top of the first page. See the image below:



Pic. 2 Resulting signed PDF file with signature field

As it can be seen from the picture, the resulting file was successfully signed using our certificate. In order to become validated, the signature certificate has to be added into the *trusted certificates list*; otherwise a yellow asterisk will appear, saying that unknown cert has been found. The signature visual representation consists of a `SignatureFieldView` linked to a corresponding `SignatureField`. It supports various visualization scenarios and makes it possible to use an image or any other supported `XObject` as signature representation.

The code

Code for capturing the signature drawing in this sample is based on MS Ink Sample and its explanation can be freely found online. Only PDF creation code is described below.

This function combines all parts together and creates resulting PDF file. First of all we create `FixedDocument` and add a new `Page` object into it. After that, we create a signature drawing (using `ClippedContent object`) and `SignatureField`. Going further we create a `SignatureFieldView` which goes to page annotations collection and provides visual representation of the field.

```
/// <summary>
/// Handles Save to PDF button click event and performs saving to PDF
/// </summary>
private async void SaveToPDFClick(object sender, RoutedEventArgs e)
{
    // create PDF document and append new page
    FixedDocument document = new FixedDocument();

    Apitron.PDF.Kit.FixedLayout.Page pdfPage = new Apitron.PDF.Kit.FixedLayout.Page(Boundaries.A4);
    document.Pages.Add(pdfPage);

    // create signature drawing based on strokes captured by ink manager
    var signatureContent = CreateSignatureDrawing();

    // register signature drawing as XObject for future reference
    document.ResourceManager.RegisterResource(new FixedContent("signatureDrawing",new
Boundary(0,0,inkPanel.Width,inkPanel.Height),signatureContent));

    // create signature field, set its settings and append to document
    SignatureField signatureField = await CreateSignatureField();
    signatureField.ViewSettings = new SignatureFieldViewSettings()
{Description =  Description.None, Graphic = Graphic.XObject, GraphicResourceID = "signatureDrawing"};
    document.AcroForm.Fields.Add(signatureField);

    // scale factor for signature
    double signatureScaleFactor = 0.3;

    // add signature view widget on PDF page, it's being linked to corresponding field and its size is
being set using scale factor defined above
    pdfPage.Annotations.Add(new SignatureFieldView(signatureField, new Boundary(0,Boundaries.A4.Top-
inkPanel.Height*signatureScaleFactor,inkPanel.Width*signatureScaleFactor,Boundaries.A4.Top)));

    // create output file
    StorageFile outputFile = await ApplicationData.Current.LocalFolder.CreateFileAsync("signature.pdf",
CreationCollisionOption.ReplaceExisting);
    // save resulting PDF to local storage, can be found at
    // C:\Users\[user name]\AppData\Local\Packages\[package name]\LocalState\signature.pdf
    using (Stream stream = await outputFile.OpenStreamForWriteAsync())
    {
        document.Save(stream);
    }
}
```

This function creates signature drawing using strokes captured by `InkManager` object. It also sets the stroking parameters e.g. stroking color and line width.

```
/// <summary>
/// Create signature drawing consisting of PDF drawing commands translated from ink manager.
/// </summary>
private ClippedContent CreateSignatureDrawing()
{
    ClippedContent signatureContent = new ClippedContent(0, 0, inkPanel.Width, inkPanel.Height);

    // set stroking parameters
    signatureContent.SetDeviceStrokingColor(RgbColors.Red.Components);
    signatureContent.SetLineWidth(STROKETHICKNESS);

    // since PDF coordinate system has inverted Y axis, we invert it here by applying transformation
    signatureContent.ModifyCurrentTransformationMatrix(1, 0, 0, -1, 0, inkPanel.Height);

    // the ink path
    Apitron.PDF.Kit.FixedLayout.Content.Path strokePath = new
Apitron.PDF.Kit.FixedLayout.Content.Path();

    // Get the InkStroke objects
    IReadOnlyList<InkStroke> inkStrokes = inkManager.GetStrokes();

    // Process each stroke
    foreach (InkStroke inkStroke in inkStrokes)
    {
        // Get the stroke segments.
        IReadOnlyList<InkStrokeRenderingSegment> segments;
        segments = inkStroke.GetRenderingSegments();

        // Process each stroke segment.
        bool first = true;
        foreach (InkStrokeRenderingSegment segment in segments)
        {
            // The first segment is the starting point for the path.
            if (first)
            {
                strokePath.MoveTo(segment.BezierControlPoint1.X, segment.BezierControlPoint1.Y);
                first = false;
            }

            // Copy each ink segment into a bezier segment.
            BezierSegment bezSegment = new BezierSegment();
            bezSegment.Point1 = segment.BezierControlPoint1;
            bezSegment.Point2 = segment.BezierControlPoint2;
            bezSegment.Point3 = segment.Position;

            // Add the bezier segment to the path.
            strokePath.AppendCubicBezier(segment.BezierControlPoint1.X, segment.BezierControlPoint1.Y,
                segment.BezierControlPoint2.X, segment.BezierControlPoint2.Y, segment.Position.X,
                segment.Position.Y);
        }
    }

    // stroke the path
    signatureContent.StrokePath(strokePath);
    return signatureContent;
}
```

A small function that loads signature certificate from application resources and creates new `SignatureField` from this data.

```
/// <summary>
/// Creates signature field by loading the corresponding certificate from app resources.
/// </summary>
private async Task<SignatureField> CreateSignatureField()
{
    SignatureField signatureField = new SignatureField("signature");

    using (Stream signatureDataStream = await
Package.Current.InstalledLocation.OpenStreamForReadAsync("data\\JohnDoe.pfx"))
    {
        signatureField.Signature = Signature.Create(new Pkcs12Store(signatureDataStream, "password"));
    }

    return signatureField;
}
```

Conclusion

Getting PDF document signed is not that hard as it can be thought from the first sight. With the right tool in hands you'll be able to do it without any problems and on many supported platforms (Windows .NET, Windows Store, Windows Phone, Xamarin Android and Xamarin iOS, Mono). Download Apitron PDF Kit library from our website (link) and read the free book written for developers by our team (get the book here). Do not hesitate to contact us if you have any questions or would like to know more about our products.