Setting custom properties for PDF signatures

Written by Apitron Documentation Team

Introduction

Software modules that digital signatures of the PDF file are called *Filters* (signature handlers) in PDF terms. There are a few existing filters e.g. Adobe.PPKLite or Adobe.PubSec.

When the signature object is being created it may have a preferred *Filter* property assigned to it. There is also an additional *SubFilter* property of the signature that defines how the signature data is being stored in the file; a few predefined values are as follows:

1. For signing PDF files using PKCS#1, the only value of SubFilter that should be used is *adbe.x509.rsa_sha1*, which uses the RSA encryption algorithm and SHA-1 digest method.

2. When PKCS#7 signatures are used allowed values are:

adbe.pkcs7.detached: The original signed message digest over the document's byte range shall be incorporated as the normal PKCS#7 SignedData field. No data shall be encapsulated in the PKCS#7 SignedData field.

adbe.pkcs7.sha1: The SHA1 digest of the document's byte range shall be encapsulated in the PKCS#7 SignedData field with ContentInfo of type Data. The digest of that SignedData shall be incorporated as the normal PKCS#7 digest.

Additional information can be found in PDF specification, see section 12.8.3 Signature Interoperability. It is intended that conforming readers allow interoperability between signature handlers; that is, a PDF file signed with a handler from one provider shall be able to be validated with a handler from a different provider.

If present, the SubFilter property of the signature shall specify the encoding of the signature value and key information, while the Filter property shall specify the preferred handler that should be used to validate the signature. When handlers are being registered according to specification they shall specify the SubFilter encodings they support enabling handlers other than the originally used to validate the signatures created.

So, to summarize all the things above: Filter - defines validation module, SubFilter – defines how to store the signature value and key info in PDF file.

In this example we'll set a few custom properties for the signature, namely the name of the application that created the signature and GPS coordinates of the signer's location.

Internally it uses the SignatureField.PropBuild dictionary to set the software module name when the signature is being created. PropBuild dictionary is a special dictionary that allows writing custom values along with the signature. Read the PDF Signature Build Dictionary Specification to find details about it.

The code

```
class Program
{
    private static void Sign(string pathToDocument, string pathToCertificate, string password,
        string pathToSignatureImage, Boundary signatureViewLocation)
    {
        // open existing document andd sign once
        using (Stream inputStream = new FileStream( pathToDocument,
            FileMode.Open, FileAccess.ReadWrite))
        {
            using (FixedDocument doc = new FixedDocument(inputStream))
            {
                string imageResourceId = Guid.NewGuid().ToString("N");
                string signatureFieldId = Guid.NewGuid().ToString("N");

                // register signature image resource
                doc.ResourceManager.RegisterResource(new Image(imageResourceId,
                    pathToSignatureImage));

                // create first signature field and initialize it using a stored certificate
                SignatureField signatureField = new SignatureField(signatureFieldId);
                using (Stream signatureDataStream = File.OpenRead(pathToCertificate))
                {
                    signatureField.Signature = new Pkcs7DetachedSignature(
                        new Pkcs12Store(signatureDataStream,password));
                    // set the software module name
                    signatureField.Signature.SoftwareModuleName =
                       "MyApp based on Apitron PDF Kit for .NET";
                    // set the GEO location of the place where the signature was created
                    signatureField.PropBuild.SetValue("GEOTAG", "38.8977° N, 77.0365° W");
                }

                // add signature fields to the document
                doc.AcroForm.Fields.Add(signatureField);

                // create first signature view using the image resource
                SignatureFieldView signatureView = new SignatureFieldView(signatureField,
                    signatureViewLocation);
                signatureView.ViewSettings.Graphic = Graphic.Image;
                signatureView.ViewSettings.GraphicResourceID = imageResourceId;
                signatureView.ViewSettings.Description = Description.None;

                // add views to page annotations collection
                doc.Pages[0].Annotations.Add(signatureView);

                // save as incremental update
                doc.Save();
            }
        }

        Process.Start("signed.pdf");
    }

    private static void CreatePDFDocument(string fileName)
    {
        using (Stream stream = File.Create(fileName))
        {
            FlowDocument doc = new FlowDocument() { Margin = new Thickness(10) };
            doc.Add(new TextBlock("Signed using Apitron PDF Kit for .NET, the signature has a custom
            property containing app name. " +
            "Click on the signature image and select \"Signature Properties...\"->\"Advanced
            Properties...\""));
            doc.Write(stream, new ResourceManager());
        }
    }
```

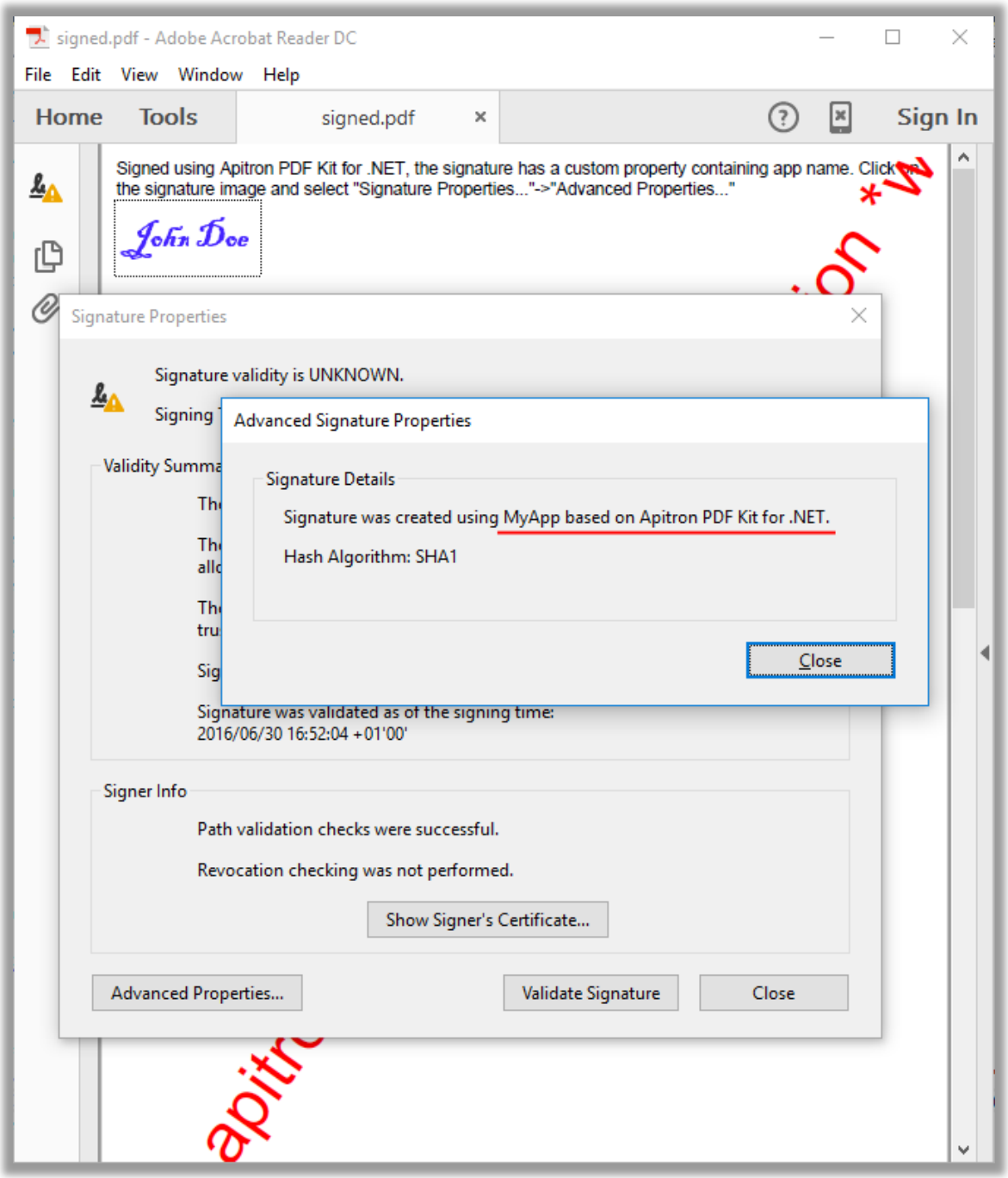
```
    static void Main(string[] args)
    {
        string fileName = "signed.pdf";

        CreatePDFDocument(fileName);

        // sign once and save
        Sign(fileName, "../../data/certs/JohnDoe.pfx", "password",
            "../../data/images/signatureImage.png", new Boundary(10, 750, 110, 800));
    }
}
```

Please note that setting custom properties using PropBuild dictionary is only possible if you use PKCS7 detached signature. The complete code sample can be found in our github repo.

And the result:



Pic. 1 Custom signature properties added using Apitron PDF Kit for .NET

Summary

As you can see, Apitron PDF Kit provides an easy to use API that allows you to solve complex tasks like signing PDF and setting custom signature properties using a couple lines of code. Additionally, its cross-platform nature simplifies development if you're targeting multiple platforms at once e.g. .NET ecosystem, Android and iOS (using Xamarin) as it's available for all modern web, desktop or mobile platforms. Contact us if you have any questions and we'll be happy to help you.