Proposed Title: PDF/A and Digital Signature – A solution to complying with Record keeping obligations in the Financial Sector.

Introduction

Record keeping is a serious business. Various laws and regulations, such as the Sarbanes-Oxley Act (SOX), and regulations from financial regulatory bodies such as the SEC, FINRA and CFTC in the US, IIROC in Canada or ESMA in the EU, place an obligation on financial services providers to maintain accurate and accessible records for a number of years. The length of time that the documents need to be retained differs between regulations but having to keep records for 7 years or longer is not unusual.

As an example of the complexity of the regulations for just one authority check out the comprehensive list of records required to be preserved by FINRA.

Typically, the regulations do not prescribe the specific technologies or formats that should be used. Instead, they merely sate that be accurately maintained and readily accessible for inspection. SEC for example via Rule 17a-4(f), explicitly states the requirements of preserving electronically stored records in a tamper-proof, non-rewritable, non-erasable format.

Getting it wrong can be expensive. Back in 2016 FINRA fined 12 firms over **14 million dollars** due to a lack of proper protection from record alterations.

In this article we will look at how two features of the Apyse SDK – PDF/A Creation and Digital Signatures - can come together to give an effective solution to document retention.

What is PDF/A?

PDF/A is a type of PDF, specifically designed (ISO 19005:1/2/3/4) for the long-term preservation of electronic documents. PDF/A differs from 'normal' PDF by prohibiting features which might cause problems with long-term archiving – such as non-embedded fonts, JavaScript, encryption and so on. The intention is that the document should remain viewable indefinitely, and importantly, should appear the same to **everyone** that looks at it.

If that sounds like regulators are just making up unnecessary rules, then think again. There can be real-life consequences of things as innocuous as a PDF using a specific font.

If, for example, a PDF contains text in a font that is not **embedded** in the PDF, then it will attempt to find a match based on the fonts that are installed on the machine where it is displayed. Exactly what is displayed then depends on the font that is substituted, and that could change from one machine to another.

If you are lucky then a good match will be found, and no-one will notice.

If you are slightly less lucky, then the characters in the substituted font might be a different size but otherwise look correct. That might look slightly odd, or it could also result in changes in the layout and flow of the document, potentially pushing text from one page onto another. You can imagine the confusion that would arise if on one machine, but not others, a table referred to as being at 'the bottom of page 5', suddenly ends up at the top of page 6.

Worse still, there might be no match found at all for a specific character. In that case the document might appear correctly on one machine, but exactly the same document might have a missing, or an entirely different, character on a different machine. You can imagine that if the missing character was something like a comma, then the meaning of the text could be altered. Initially the meaning of the document might still be understood, since people will remember what it was about. But in five years' time, when staff have moved on, that might no longer be possible. And commas also matter – in one Canadian case, the presence of a comma changed the meaning of a contract, costing one of the companies a million dollars.

Imagine how complex, and costly that case would have been if the comma was there only when viewed on some computers.

Clearly there is a very real problem. Thankfully the Apryse family offer several ways to **create** PDF/A compliant documents.

iText can be used to create PDF/A compliant documents as part of a document generation and archiving process.

The Fluent document generation system, which takes templates created in Office and populates them with data from any of a huge range of data source to create documents, including PDF/A compliant ones.

However, if you already have documents that are not PDF/A, whether they are just 'normal' PDF, Office documents, scanned files or images, then conversion to PDF/A, rather than generation, is required. Thankfully Apryse can still help you using **Qoppa PDF Studio Pro**, the **Apryse SDK**, or **WebViewer** either as part of your own website, or hosted as **xodo.com**.

Converting to PDF/A using PDF Studio Pro

Qoppa PDF Studio Pro has a straightforward UI that leads you through the conversion of a PDF into a PDF/A document. It informs you that certain aspects of the PDF may be altered, and you should make sure that those changes will not be removing important information.

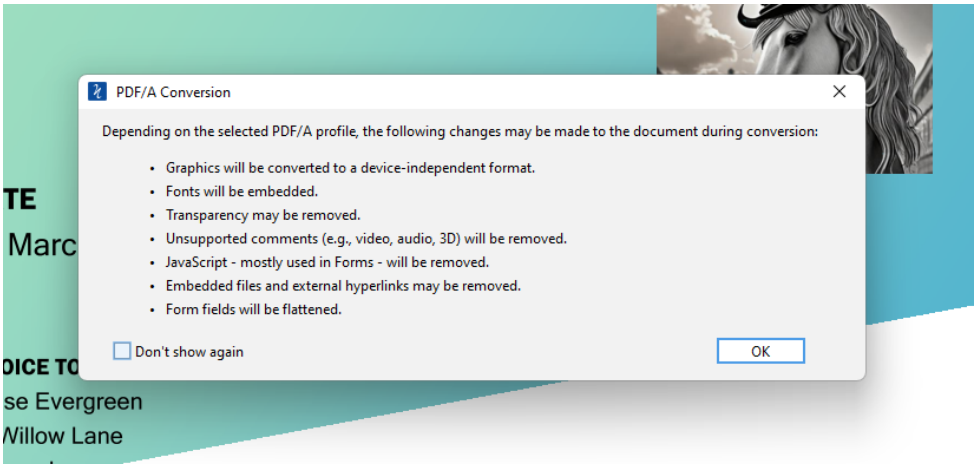

Figure 1 - The UI in PDF Studio Pro when converting a PDF invoice to PDF/A-2b. The user is informed of what will happen.

Once the conversion succeeds the PDF will be shown, indicating that the document is tagged as PDF/A compliant. If you wish you can also get the software to verify compliance.

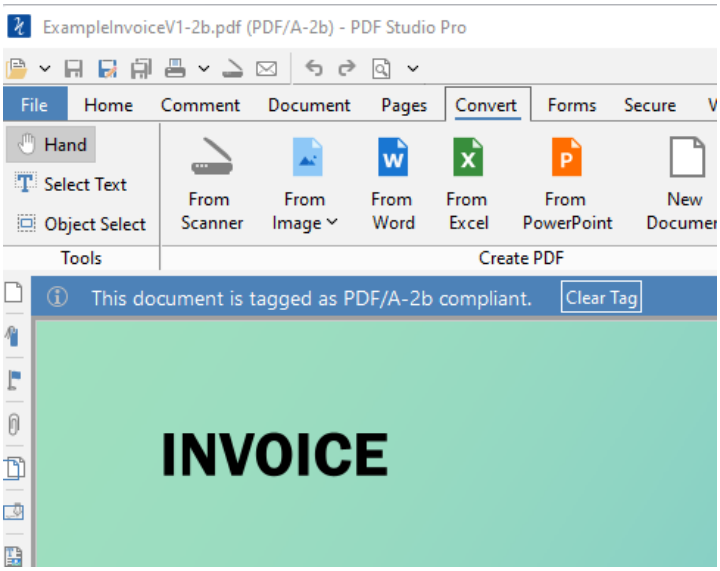

Figure 2 - A PDF/A compliant document within PDF Studio Pro - it is clear that the PDF has been tagged as being PDF/A compliant.

Converting a PDF to PDF/A using the Apryse SDK.

The Apryse SDK ships with large number of samples to make life easy. You can also access the sample projects, which are available in a large number of programming languages, directly from the documentation website.

At its simplest, conversion to PDF/A just needs a few lines of code. In the following snippet the conversion is to PDF/A-2b, although all of the conformance levels including **4, 4e and 4f** are available.

[Start code snippet – C#]

```
                string sourceFile = "ExampleInvoiceV1.pdf";
                string outputFile = "ExampleInvoiceV1-pdfa.pdf";
                using (PDFACompliance pdf_a = new PDFACompliance(true, input_path +
sourceFile, null, PDFACompliance.Conformance.e_Level2B, null, 10, false))
                {

                    pdf_a.SaveAs(output_path + outputFile, false);
                }
```

`[End code snippet]`

I've not included all of the code for the full sample, but it also reports any PDF/A violations in the converted file, so you can be sure that the conversion succeeded.

```
PDFNet is running in demo mode.
Package: archive
ExampleInvoiceV1.pdf is NOT a valid PDFA.
 - e_PDFA 7114: Invalid PDF/A part number.
   Objects: 30

ExampleInvoiceV1-pdfa.pdf: OK.
```

Figure 3 - Sample output, also showing that the initial file was not PDF/Complaint, but that the generated one is.

Converting to PDF/A using xodo.com

Xodo.com allows you to convert from PDF to PDF/A in your web browser, and as with the Apryse SDK, all of the variants including **4, 4e and 4f** are available.

All you need to do is drag the PDF into the browser, choose the PDF/A type, and press **Convert**.

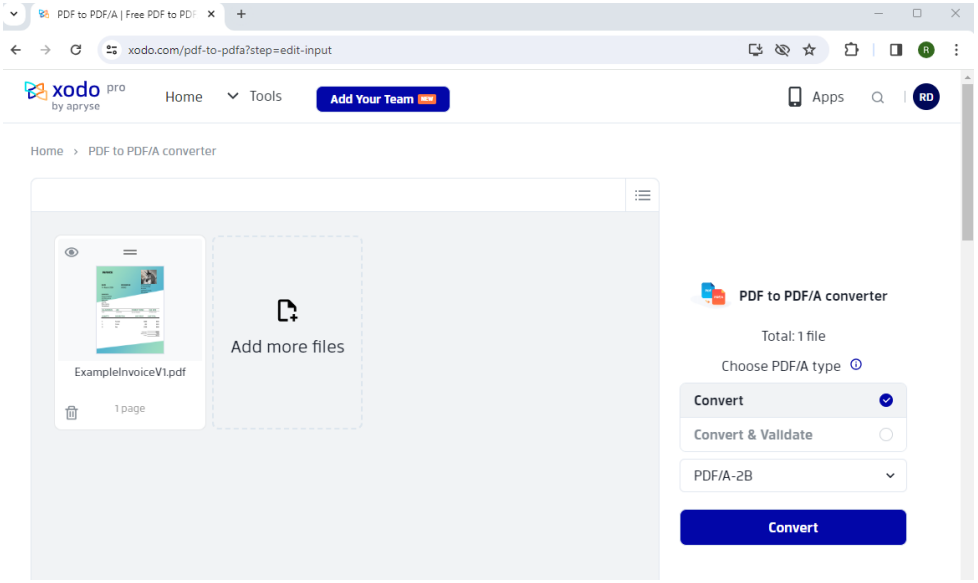

Figure 4 - Xodo.com indicating that the file was successfully converted to PDF/A.

After a few seconds the conversion will complete, and you can save your file.

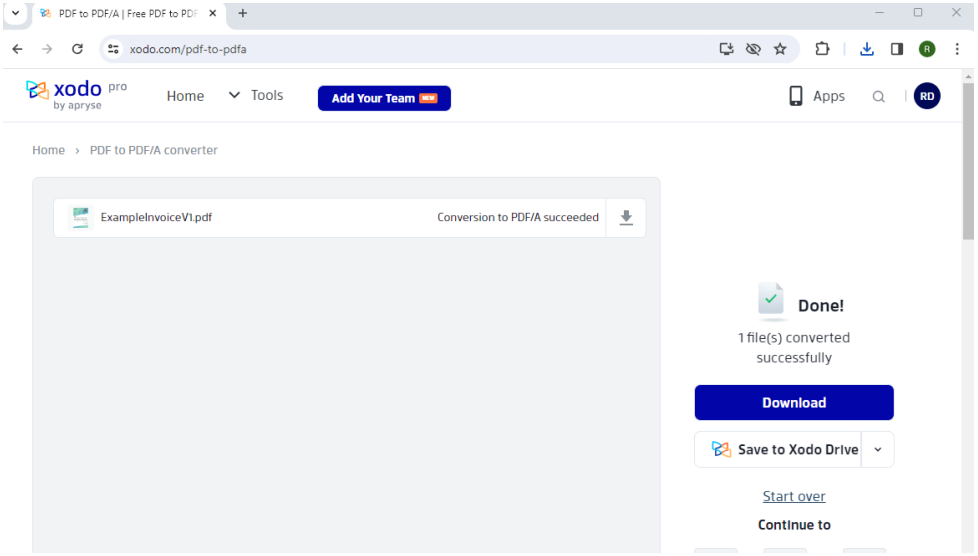

Figure 5 - Xodo.com, showing the UI for converting to PDF/A

WebViewer (which is doing much of the work in Xodo) also supports converting from PDF to PDF/A. You can try that out for yourself using the Apryse Showcase.

For more information check out converting files to PDF/A using Apryse.

Being able to store documents long term is only part of the solution. How can you be certain that the document originated with the person that you think? This is where digital signatures can help.

What are Digital Signatures?

The term '**digital signatures**' can be confusing because it has been used for more than one purpose.

1. a visible representation of a handwritten signature – an **eSignature -** which might be a scanned image of a wet signature, or it might be the person's name in a fancy font, or drawn using a mouse. When applied using some software products such as "**Xodosign**" (or more correctly "**xodo sign**") these signatures are an efficient and legal way of getting documents signed in many jurisdictions (and are legislated for example, via the ESIGN Act in the US).

   On the other hand, a homegrown software solution that is implemented badly could create signatures that, from a casual inspection, look valid but in reality are just an image within the document which can easily be repudiated.

2. a way of identifying, beyond repudiation, via the means of AES (essentially a self-signed certificate) and QES (where the certificate is issued by a trusted Certificate Authority), that the document was signed by the person that claims to have signed it. Furthermore, digital signing can be used to prove that the document has not been subsequently modified. At the nuts and bolts level, there are various technologies that support this, for example PadES, which is designed specifically for use with PDFs.

[start call out]

It's a complicated subject, and you may wish to read more about Understanding Digital Signatures

[End call out]

Apryse and Digital Signatures

Just as with PDF/A conversion there are multiple ways within the Apryse family of adding a digital signature. We will look at:

1. Qoppa PDF Studio Pro
2. Apryse SDK.

We won't look at Xodo Sign in this article though, since it deserves an article all of its own, but if you are inpatient, head over to the help center. Suffice to say, it is a fully featured system that allows you to securely approve, send and sign documents online. It has a great and intuitive Web UI, and can also if need be accessed via a secure API.

PDF Studio Pro.

PDF Studio Pro makes signing a document a breeze. Just open the document, choose **Secure** then click on **Digitally Sign**.

You will need to choose a location where the signature will be shown, and you are able to specify details for the signature.

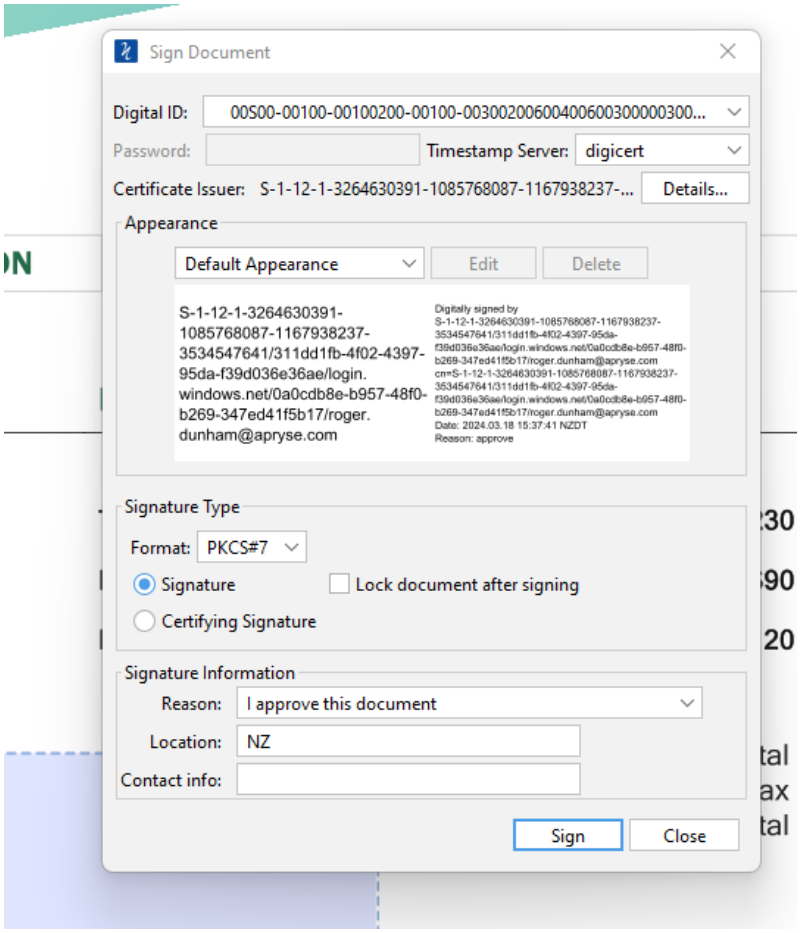

Figure 6 - Adding a digital signature in PDF Studio Pro.

Once the document is signed, and saved, PDF Studio Pro will let you know that the document is signed and will verify (or not!) that the signature is valid.

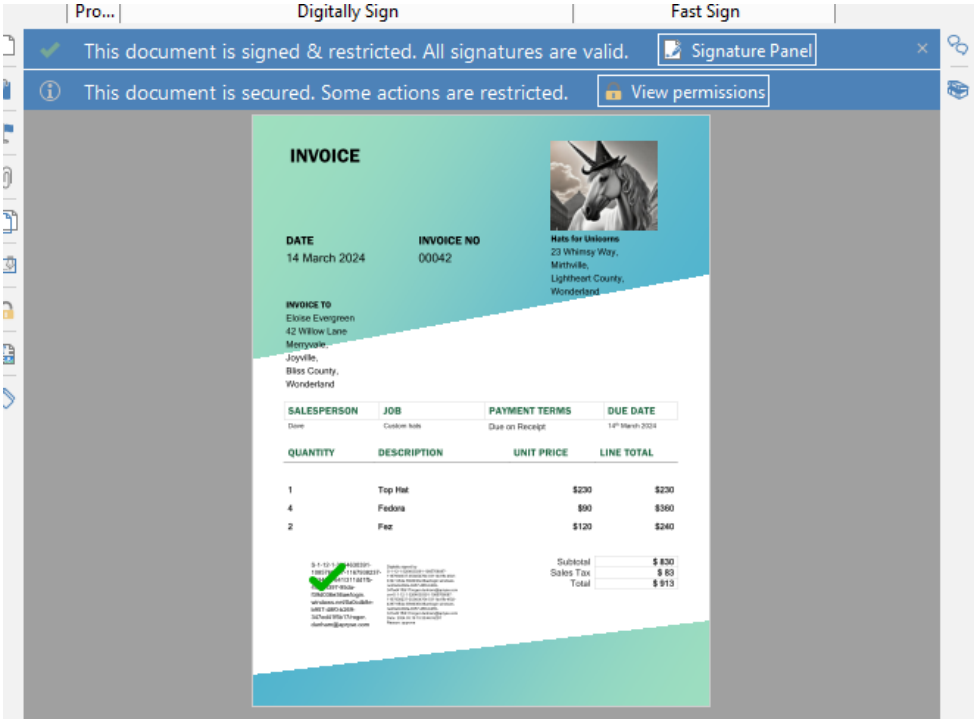

Figure 7 - A digitally signed document in PDF Studio Pro. The blue banner, and the green tick mark in the signature box indicate that the signature is valid.

Signing documents with the Apryse SDK

The Apryse SDK can also digitally sign documents.

As we saw with PDF/A, the Apryse SDK comes with lots of sample code, and this includes the ability to sign documents. There is a great, and well documented sample available which is included when you download the Apryse SDK, or alternatively you can download it expliticly in a wide range of

programming languages from https://docs.apryse.com/documentation/samples.

Figure 8 - The range of programming languages that support the Digital Signature sample.

The sample code is a great place to get started, and performs various signing and signature verification test.

The following snippet is based on (but simplified) from the method **`CertifyPDF`**

[Start code snippet – C#]

```
                DigitalSignatureField certification_sig_field =
doc.CreateDigitalSignatureField(in_cert_field_name);
                SignatureWidget widgetAnnot = SignatureWidget.Create(doc, new Rect(143, 150,
219, 170), certification_sig_field);
                page1.AnnotPushBack(widgetAnnot);

                // (OPTIONAL) Add an appearance to the signature field.
                Image img = Image.Create(doc, in_appearance_image_path);
                            widgetAnnot.CreateSignatureAppearance(img);

                            // Prepare the document locking permission level. It will be
applied upon document certification.
                            Console.Out.WriteLine("Adding document permissions.");

      certification_sig_field.SetDocumentPermissions(DigitalSignatureField.DocumentPermission
s.e_annotating_formfilling_signing_allowed);

certification_sig_field.CertifyOnNextSave(in_private_key_file_path, in_keyfile_password);
```

[End code snippet]

The code programmatically adds a new signature field (at the location 143, 15, 219, 170), places an image at that location, sets various document properties, then certifies the document using a public certificate key and its password.

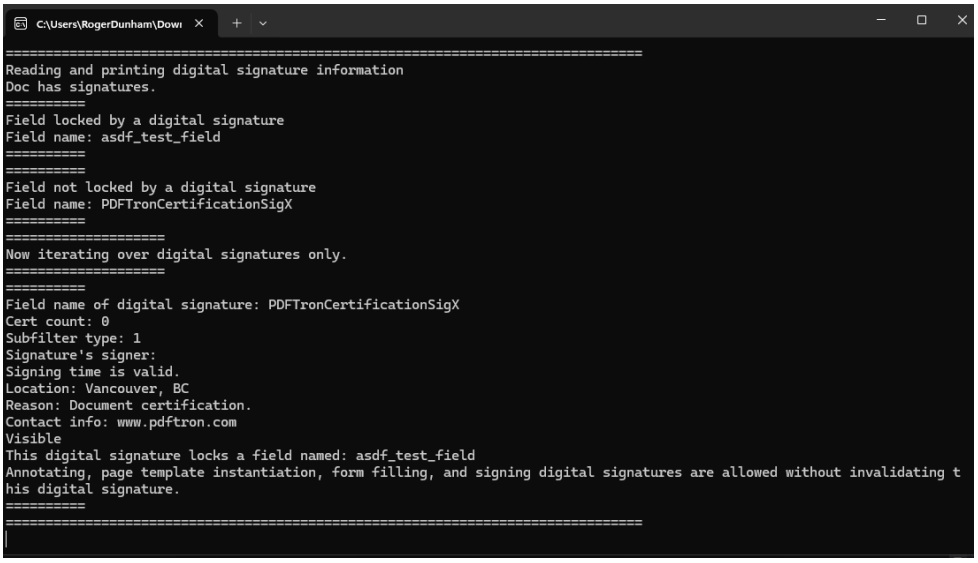

Figure 9 - Output from the sample code (which differs from the snippet above), which also verifies that the signature is valid.

The sample code ships with a suitable key file - pdftron.pfx – but for production use, you will need to get you own one. Other signing technologies, including custom signature handlers can also be used.

If you would rather support the ability to sign documents within the browser from within your own application, then WebViewer supports this, and you can check that out using the Apryse Showcase.

The natural order of things

If you want to store documents that are both PDF/A compliant and digitally signed, then the order in which those processes are preformed matters.

Digitally signing a document means that it will be possible to identify if changes were made after the signature was applied. Sometimes that is valid (for example annotations can potentially added if that was specified when the document was signed), but often changes will invalidate the signature – which, after all, is the entire point of having the document signed.

This means that the document should be converted to PDF/A before it is signed, since doing so afterwards is likely to invalidate the signature.

Wrapping up

PDF/A is great for archiving data in a way that is designed to be readable in years, or decades, to come.

Furthermore, Digital signatures, particularly if backed with a certificate are a great way to be able to verify the origin and validity of a document.

Combined together these provide a strategy for securely archiving documents in order to comply with numerous Acts and Regulations.

Apryse has your back. The Apryse family provides ways to generate, or convert, documents to PDF/A then digitally sign, and verify those signatures. And this can be done within a desktop app, a web-based solution, or a server-based API.

There is a wealth of documentation available for both of these techniques, as well as for the other functionality that Apryse offers. While PDF/A and Digital signatures may be today's only requirements, Apryse can also help when your requirements expand, perhaps to redaction or the need to support multiple users annotating the same document.

As a final word, the subjects covered in this article *are* complex, so feel free to reach out to us on Discord if you need any further assistance.

Commented [RD1]: This link may need updating