FREE eBook

LEARNING

pdf

Free unaffiliated eBook created from
Stack Overflow contributors.

#pdf

Table of Contents

About 1

Chapter 1: Getting started with pdf 2

Remarks 2

Versions 2

Examples 3

Installation or Setup 3

Code sample from pdfsharp.net 3

PDFTK Server for pdf manipulation 3

Chapter 2: Integrated PDF signatures 6

Remarks 6

Examples 6

How integrated PDF signatures are "integrated" 6

Multiple signatures in a single PDF 6

PDF signature types 7

Allowed and disallowed changes to a signed document 7

Allowed actions for certified documents 7

Certified with no changes allowed 7

Allowed 8

Disallowed 8

Certified with form fill-in and digital signatures allowed 8

Allowed 8

Disallowed 8

Certified with annotations, form fill-in, and digital signatures, allowed 8

Allowed 8

Disallowed 8

Allowed actions for signed but uncertified documents 8

Allowed 9

Disallowed 9

The signed byte range 9

Interoperable signature types 9

Credits.....11

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: pdf

It is an unofficial and free pdf ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official pdf.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with pdf

Remarks

Portable Document Format (PDF) is a file format used to present and exchange documents reliably, independent of software, hardware, or operating system. Invented by Adobe, PDF is now an open standard maintained by the International Organization for Standardization (ISO). PDFs can contain images, links, buttons, form fields, audio, video, and business logic. They can also be signed electronically, commented on and encrypted and are easily viewed using free Acrobat Reader DC software. They are also viewable using Google Drive and other software.

PDF files can be created specifically to be accessible for people with disabilities. PDF file formats in use as of 2014 can include tags (XML), text equivalents, captions, audio descriptions, etc. Tagged PDF is required in the PDF/A-1a specification. Some software can automatically produce tagged PDFs, but this feature is not always enabled by default. Some screen readers, including JAWS, Window-Eyes, Hal, and Kurzweil 1000 and 3000 can read tagged PDFs aloud, as can later versions of the Acrobat and Acrobat Reader programs. Moreover, tagged PDFs can be re-flowed and magnified for readers with visual impairments.

Problems remain with adding tags to older PDFs and those that are generated from scanned documents. In these cases, accessibility tags and re-flowing are unavailable, and must be created either manually or with OCR techniques. These processes are inaccessible to some people with disabilities.

Source

Versions

Version	Software(s)	Release Date
1.0	Adobe Acrobat 1.0	1993-06-01
1.1	Adobe Acrobat 2.0	1994-11-01
1.2	Adobe Acrobat 3.0	1996-11-01
1.3	Adobe Acrobat 4.0	1999-04-01
1.4	Adobe Acrobat 5.0	2001-05-01
1.5	Adobe Acrobat 6.0, Adobe Reader 6.0	2003-04-01
1.6	Adobe Acrobat 7.0, Adobe Reader 7.0	2005-01-01
1.7	Adobe Acrobat 8.0, Adobe Reader 8.0	2006-10-01

Examples

Installation or Setup

To view a pdf you can download Adobe reader for free . You can create pdfs programmatically with the help of, e.g by using iTextSharp, jsPDF or PDFSharp (there are other libraries available)

Code sample from pdfsharp.net

Code source View the output here

```
using System;
using System.Diagnostics;
using System.IO;
using PdfSharp;
using PdfSharp.Drawing;
using PdfSharp.Pdf;
using PdfSharp.Pdf.IO;

namespace HelloWorld
{
  /// <summary>
  /// This sample is the obligatory Hello World program.
  /// </summary>
  class Program
  {
    static void Main(string[] args)
    {
      // Create a new PDF document
      PdfDocument document = new PdfDocument();
      document.Info.Title = "Created with PDFsharp";

      // Create an empty page
      PdfPage page = document.AddPage();

      // Get an XGraphics object for drawing
      XGraphics gfx = XGraphics.FromPdfPage(page);

      // Create a font
      XFont font = new XFont("Verdana", 20, XFontStyle.BoldItalic);

      // Draw the text
      gfx.DrawString("Hello, World!", font, XBrushes.Black,
        new XRect(0, 0, page.Width, page.Height),
        XStringFormats.Center);

      // Save the document...
      const string filename = "HelloWorld.pdf";
      document.Save(filename);
      // ...and start a viewer.
      Process.Start(filename);
    }
  }
}
```

PDFTK Server for pdf manipulation

Install PDFTK Server from https://www.pdflabs.com/tools/pdftk-server/

PDFtk Server is a command line tool which can:

```
•    Merge PDF Documents or Collate PDF Page Scans
•    Split PDF Pages into a New Document
•    Rotate PDF Documents or Pages
•    Decrypt Input as Necessary (Password Required)
•    Encrypt Output as Desired
•    Fill PDF Forms with X/FDF Data and/or Flatten Forms
•    Generate FDF Data Stencils from PDF Forms
•    Apply a Background Watermark or a Foreground Stamp
•    Report PDF Metrics, Bookmarks and Metadata
•    Add/Update PDF Bookmarks or Metadata
•    Attach Files to PDF Pages or the PDF Document
•    Unpack PDF Attachments
•    Burst a PDF Document into Single Pages
•    Uncompress and Re-Compress Page Streams
•    Repair Corrupted PDF (Where Possible)
```

PDFtk Server does not require Adobe Acrobat or Reader, and it runs on Windows, Mac OS X and Linux.

Collate scanned pages

```
pdftk A=even.pdf B=odd.pdf shuffle A B output collated.pdf
```

or if odd.pdf is in reverse order:

```
pdftk A=even.pdf B=odd.pdf shuffle A Bend-1 output collated.pdf
```

Decrypt a PDF

```
pdftk secured.pdf input_pw foopass output unsecured.pdf
```

Encrypt a PDF using 128-bit strength (the default), withhold all permissions (the default)

```
pdftk 1.pdf output 1.128.pdf owner_pw foopass
```

Same as above, except password baz must also be used to open output PDF

```
pdftk 1.pdf output 1.128.pdf owner_pw foo user_pw baz
```

Same as above, except printing is allowed (once the PDF is open)

```
pdftk 1.pdf output 1.128.pdf owner_pw foo user_pw baz allow printing
```

Join in1.pdf and in2.pdf into a new PDF, out1.pdf

```
pdftk in1.pdf in2.pdf cat output out1.pdf
```

or (using handles):

```
pdftk A=in1.pdf B=in2.pdf cat A B output out1.pdf
```

or (using wildcards):

```
pdftk *.pdf cat output combined.pdf
```

Remove page 13 from in1.pdf to create out1.pdf

```
pdftk in.pdf cat 1-12 14-end output out1.pdf
```

or:

```
pdftk A=in1.pdf cat A1-12 A14-end output out1.pdf
```

Read Getting started with pdf online: https://riptutorial.com/pdf/topic/3062/getting-started-with-pdf

Chapter 2: Integrated PDF signatures

Remarks

Integrated PDF signatures are explained quite graphically and in more detail in the Adobe document Digital Signatures in a PDF. They furthermore are specified in the PDF specification ISO 32000-1:2008 made available here by Adobe in section 12.8 *Digital Signatures.*

Depending on your programming context, there are many PDF libraries supporting the creation of integrated PDF signatures and also many products using these libraries. Some of them are even available for free subject e.g. to the AGPL.

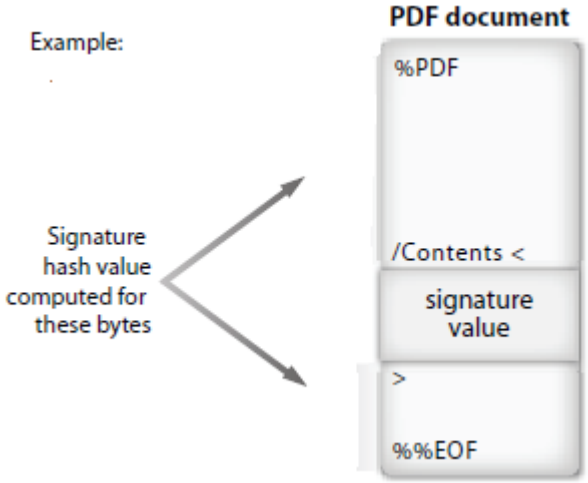
(This article is essentially a copy of this answer on the information security site, this answer on stackoverflow, and some words from the PDF specification.)

Examples

How integrated PDF signatures are "integrated"

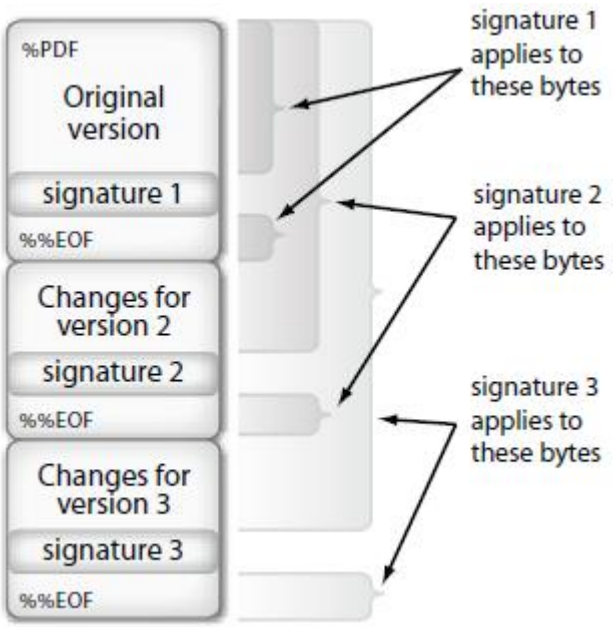
You want to add a signature to a PDF in a way that a standard conform PDF viewer (e.g. Adobe Reader) will recognize, display, and validate as an integrated PDF signature.

In that case you cannot simply externally create a signature covering the original PDF as is and expect to now have to merely somehow append that signature to the file. Instead you first have to build a new revision of the PDF document which includes a PDF AcroForm signature field whose value is a signature dictionary whose **/Contents** entry will eventually hold the signature of the whole new revision with the exception of the **/Contents** entry contents.



Multiple signatures in a single PDF

If multiple signatures are to be integrated into a PDF, this is done by means of incremental PDF updates (explicitly **not** by adding multiple SignerInfo structures to a single integrated CMS signature container!):



The cryptographic verification of these signatures merely guarantees that the byte range signed by the respective signature has not been tampered with. If does not guarantee that PDF objects defined in that range have not been replaced with other ones in any newer revision.

Checking in what way the additions in later revisions change the appearance of the content signed by an earlier signature is a separate verification task which is not trivial.

PDF signature types

A PDF document may contain the following standard types of signatures:

- Any number of **approval signatures** in signature form fields.
- At most one **certification signature** in a signature form field. It enables the author to specify what changes shall be permitted to be made to the document and what changes invalidate the author's signature, cf. *"Allowed and disallowed changes to a signed document"* below.
- At most two **usage rights signatures** referenced from the PDF's permissions dictionary. Usage rights signatures shall be used to enable additional interactive features that may not be available by default in a reader. These signatures usually are not presented to the user but merely evaluated by the PDF reader program.

Allowed and disallowed changes to a signed document

In the Adobe technical white paper Adobe Acrobat 9 Digital Signatures, Changes and Improvements, especially its section "Allowed and disallowed changes", Adobe *clarifies the allowed changes* (as seen by Acrobat 9 and up) *that can be made to a certified or signed document without invalidating the signatures applied to the document.*

Allowed actions for certified documents

Certified with no changes allowed

Allowed

- No changes allowed

Disallowed

- Digitally signing
- Supplying form field values
- Adding or editing annotations
- Adding form fields
- Changing page content

Certified with form fill-in and digital signatures allowed

Allowed

- Supplying form field values
- Digitally signing

Disallowed

- Adding or editing annotations
- Adding form fields
- Changing page content

Certified with annotations, form fill-in, and digital signatures, allowed

Allowed

- Adding or editing annotations
- Supplying form field values
- Digitally signing

Disallowed

- Adding form fields
- Changing page content

Allowed actions for signed but uncertified documents

Allowed

- Adding signature fields (see *Limitations on adding signature fields to signed but uncertified documents*)
- Adding or editing annotations
- Supplying form field values
- Digitally signing

Disallowed

- Adding form fields other than signature fields
- Changing page content

(Even though not explicitly mentioned here, instantiating page templates most likely also is allowed whenever form fill-ins are allowed as that would conform to the PDF standard, cf. ISO 32000-1 section 12.8.2.2.2.)

The signed byte range

The specification says:

> A byte range digest shall be computed over a range of bytes in the file, that shall be indicated by the ByteRange entry in the signature dictionary. **This range should be the entire file, including the signature dictionary but excluding the signature value itself (the Contents entry). Other ranges may be used but since they do not check for all changes to the document, their use is not recommended.**

This seems to allow that you first create a signature for the original PDF and then append a new revision holding that signature indicating that range of signed bytes only contains that original revision, not the extended revision without only the signature.

In reality, though, PDF viewers (especially Adobe Reader) will only accept signatures which follow the recommendation that the signed *range should be the entire file, including the signature dictionary but excluding the signature value itself.*

Newer specifications, e.g. the ETSI PAdES specification ETSI TS 102 778 (cf. section 5.1 item b in part 2 and section 4.2 item c in part 3) even make this recommendation officially a requirements, and so will ISO 32000-2.

Interoperable signature types

As the header already says, the following list contains "interoperable signature types" which are more or less strictly defined. The PDF specification specifies a way to also include completely custom signing schemes. But let us assume we are in an interoperable situation. The the collection of signature types burns down to:

- **adbe.x509.rsa_sha1** defined in ISO 32000-1 section 12.8.3.2 *PKCS#1 Signatures*; the signature value **Contents** contain a *DER-encoded PKCS#1 binary data object*; this data

object is a fairly naked signature, in case of RSA an encrypted structure containing the padded document hash and the hash algorithm.

- **adbe.pkcs7.sha1** defined in ISO 32000-1 section 12.8.3.3 *PKCS#7 Signatures*; the signature value **Contents** contain a *DER-encoded PKCS#7 binary data object*; this data object is a big container object which can also contain meta-information, e.g. it may contain certificates for building certificate chains, revocation information for certificate revocation checks, digital time stamps to fix the signing time, ... *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.*

- **adbe.pkcs7.detached** defined in ISO 32000-1 section 12.8.3.3 *PKCS#7 Signatures*; the signature value **Contents** contain a *DER-encoded PKCS#7 binary data object*, see above. *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.*

- **ETSI.CAdES.detached** defined in ETSI TS 102 778-3 and will become integrated in ISO 32000-2; the signature value **Contents** contain a *DER-encoded SignedData object as specified in CMS*; CMS signature containers are close relatives to PKCS#7 signature containers, see above. This essentially is a differently profiled and stricter defined variant of adbe.pkcs7.detached.

- **ETSI.RFC3161** defined in ETSI TS 102 778-4 and will become integrated in ISO 32000-2; the signature value **Contents** contain a *TimeStampToken as specified in RFC 3161*; time stamp tokens again are a close relative to PKCS#7 signature containers, see above, but they contain a special data sub-structure harboring the document hash, the time of the stamp creation, and information on the issuing time server.

I would propose studying the specifications I named and the documents referenced from there, mostly RFCs. Based on that knowledge you can easily find the appropriate BouncyCastle classes to analyze the different signature **Contents**.

(Copied from this answer)

Read Integrated PDF signatures online: https://riptutorial.com/pdf/topic/5161/integrated-pdf-signatures

Credits

S. No	Chapters	Contributors
1	Getting started with pdf	Community, karel, Kurt Pfeifle, Rachel Gallen, Ray, SajithP, yms
2	Integrated PDF signatures	mkl