PDF PDF PDF

2016-08-28, BarcampGR11

Adam Tauno Williams <awilliam@whitemice.org>

Copyright

If you find this document useful or further it's distribution we would appreciate you letting us know. I also maintain an Amazon wish list for the above e-mail address.

whoami

Adam Tauno Williams (awilliam@whitemice.org)

GPG: D95ED393

Systems & Network Administrator, Morrison Industries

Active Directory, PostgreSQL, Informix, RabbitMQ, AcuCOBOL, WAN (Fiber, AT&T ASE, MPLS, Cisco IOS, IPSec), VM-ware, ISC Bind DNS & ISC DHCP, ZenOSS, AIX, CUPS/IPP, Cyrus IMAP, SASL, Postfix, ...

Meeting Coordinator, Michigan Association of Railroad Passengers

Developer (Python). OpenGropware Coils

http://www.whitemice.org
http://www.whitemiceconsulting.com
http://www.opengroupware.us

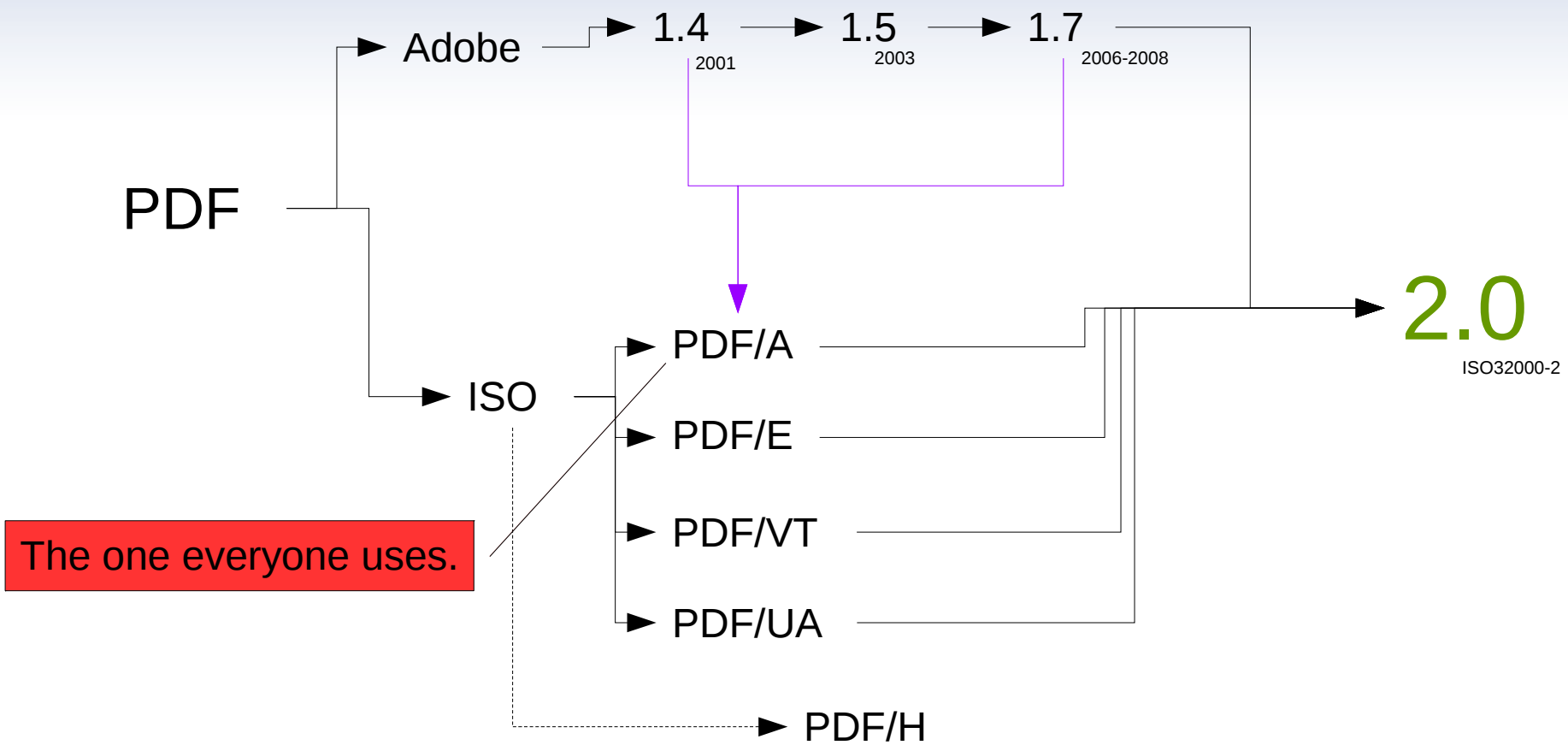
What is a PDF?

- Self-Contained!
  - May contain resources [attachments]
- Standardized
- Fixed layout
- Subordinate variant of Postscript
  - PS is global, PDF is page scope
  - PDF supports transparency
  - Additional Meta-data & Security features

Who is PDF

- Patented by Adobe
- Created 1993
- Open since 2008
  - ISO32000-1
- Split Hierarchy
  - Open – ISO
  - Proprietary – Adobe
    - Features migrate from proprietary versions to open versions.

Timeline


PDF
Adobe
1.4
2001
1.5
2003
1.7
2006-2008
2.0
ISO32000-2
ISO
PDF/A
PDF/E
PDF/VT
PDF/UA
PDF/H
The one everyone uses.

Versions

- **PDF 1.4**
  - 40-128b RC4 encryption
  - JBIG2
  - FDF w/digital signatures
  - Includes
  - Meta-data / XMP
- **PDF 1.5**
  - Usage Rights, FDF/XDF
- **PDF 1.7**
  - 256b AES
  - Video

FDF vs. XMP vs. Meta

- PDFs have no less than three mechanisms for storing meta-data
  - Meta-data tags : old school.
  - FDF : Field Definition Format
  - XMP : eXtensible Meta-Data

PyPDF2

https://pypi.python.org/pypi/PyPDF2

PDF Manipulation

- License: BSD
- Pure Python
- Features
  - meta-data
  - page operations
    - crop, merge, split
  - encryption/decryption

PyPDF2: Meta-Data

```
>>> rfile = open('nvm4_reference_guide.pdf', 'rb')
>>> from PyPDF2 import PdfFileReader
>>> reader = PdfFileReader(rfile)
>>> xmp = reader.getXmpMetadata()
>>> xmp
>>> reader.getDocumentInfo()
{'/Title': u'NVM Ref Book', '/Creator': u'FrameMaker+SGML 5.5.3',
'/Producer': u'Acrobat Distiller 3.0 for Macintosh', '/CreationDate':
u'D:19981124085503', '/Author': u'Patrick Chatterton'}
```

No XMP data.

But there is tag data.

PyPDF2: XMP

```
>>> rfile = open('IRS Electronic Signature Requirements.pdf', 'rb')
>>> from PyPDF2 import PdfFileReader
>>> reader = PdfFileReader(rfile)
>>> xmp = reader.getXmpMetadata()
>>> xmp.pdf_keywords
>>> xmp.xmpmm_documentId
u'uuid:966f0bbd-d8a8-4edf-8298-f9ca6f4d9db6'
```

May be None

XMP data is an object with properties.

PyPDF2: FDF

```
>>> rfile = open('tests/assetts/PDF-Filled-Form.pdf', 'rb')
>>> from PyPDF2 import PdfFileReader
>>> reader = PdfFileReader(rfile)
>>> pprint.pprint(reader.getFields())
>>> import pprint
>>> pprint.pprint(reader.getFields()
... )
{ u'Date': {'/AA': {'/F': IndirectObject(424, 0),
                    '/K': IndirectObject(425, 0)},
            '/FT': '/Tx',
            '/Ff': 2,
            '/T': u'Date',
            '/TU': u'Date MM/DD/YY (required)',
            '/V': u'4/10/14'},
 u'Driver': {'/FT': '/Tx',
             '/Ff': 4194306,
             '/T': u'Driver',
             '/TU': u'Driver (required)',
             '/V': u'John Doe'},
```

writer.updatePageFormFieldValues

PyPDF2: Bursting

```
signal.signal(signal.SIGALRM, timeout_alarm_handler)
signal.alarm(15)
reader = PdfFileReader(rfile, strict=False, )

writer = PdfFileWriter()
for page in range(3, 5):
    try:
        signal.alarm(15)
        writer.addPage(reader.getPage(page))
        signal.alarm(0)
    except TimeOutAlarm:
        raise BurstingTimeOutException
     else:
         writer.write(wfile)
writer = None
wfile.flush()
```

cloneDocumentFromReader

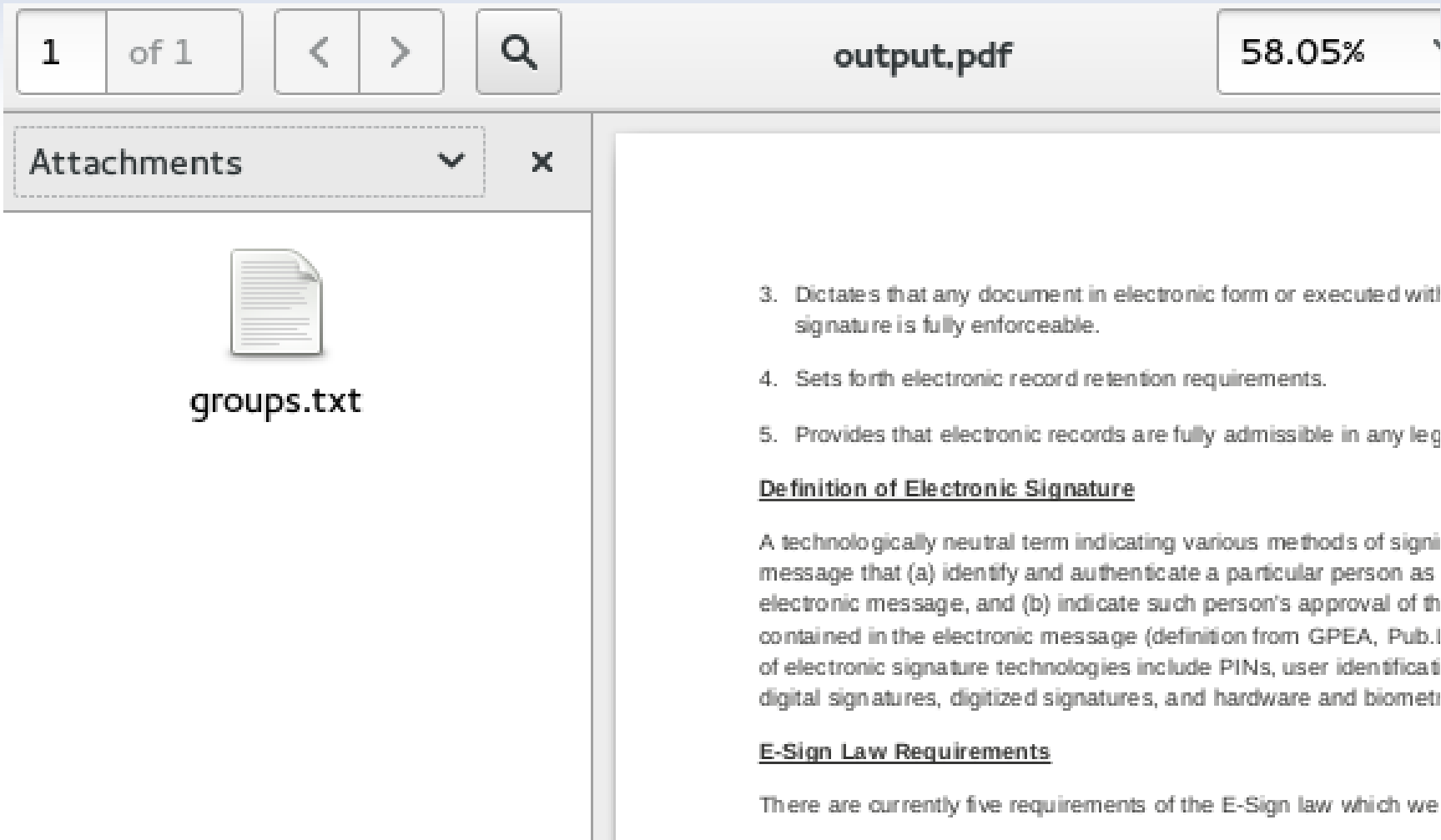
PyPDF2: Rotate Pages

```
import PyPDF2

reader = PdfFileReader(rfile)
writer = PdfFileWriter()

for pagenum in range(reader.numPages):
    page = reader.getPage(pagenum)
    page.rotateClockwise(180)
    writer.addPage(page)

wfile = open('rotated.pdf', 'wb')
writer.write(pdf_out)
wfile.close()
rfile.close()
```

output.addPage(input.getPage(1).rotate**Counter**Clockwise(90))

PyPDF2: Attachments

writer.addAttachment('groups.txt', open('/etc/group', 'rb').read())



PyPDF2: More Content

- addJS : Javascript
- addLink: add internal link
- addBookmark

PyPDF2: Extract Text

```
reader = PdfFileReader(rfile)
wfile = open('output.txt', 'wb')
for i in range(0, reader.numPage):
    page = reader.getPage(i)
    wfile.write(page.extractText())
    wfile.write('\n')
```

PDFs are not actually linear; the text may not match the *flow* of the read document.

z3c.rml

https://pypi.python.org/pypi/z3c.rml

- License: ZPL

PDF Creation

- Implementation of RML
  - AKA: A free version of ReportLab
- Robust PDF creation.

http://www.reportlab.com/software/opensource/samples/
http://www.reportlab.com/documentation/rml-samples/
http://www.reportlab.com/software/rml-reference/

RML

```
   <xsl:template match="/workorders">
      <document filename="example_3.pdf">
         <template pagesize="letter" leftMargin="20" showBoundary="0">
            <pageTemplate id="main">
               <pageGraphics>
                  <image x="11mm" y="248mm" width="5.2cm" height="1.9cm">
                    <xsl:attribute name="file">text-x-python.png</xsl:attribute>
                  </image>
                  <stroke color="gainsboro"/>
                  <drawString x="90mm" y="261mm">Workorder
Summary</drawString>
...
```

Easy and fast to generate by XSLT transform or using ElementFlow.

Creation

```
from z3c.rml import document as RMLDocument
from z3c.rml import interfaces
zope.interface.moduleProvides(interfaces.IRML2PDF)

class RMLToPDFWriter(object):

    def __init__(self, rfile):
        root = etree.parse(rfile).getroot()
        self.doc = RMLDocument.Document(root)

    def write(self, wfile):
        self.doc.process(wfile)

    def close(self):
        self.doc = None

writer = RMLToPDFWriter(rfile)
writer.write(self.wfile)
writer.close()
```

Metapdf

https://pypi.python.org/pypi/metapdf/0.3.2

PDF Data Extraction

- License: MIT
- Optimized for one task.
  - Meta data for a PDF is stored at the end of the file.
  - 50-60% performance improvement for large documents.

Metapdf

```
>>> import os
>>> from metapdf import MetaPdfReader
>>> MetaPdfReader().read_metadata(read('1984.pdf', 'rb'))
{
    '/Author': 'George Orwell',
    '/Title': '1984',
    '/ModDate': u"D:20041013081643-04'00'",
    '/CreationDate': u'D:20001221232522Z',
    '/Producer': u'Acrobat PDFWriter 4.0 for Windows',
    '/Creator': u'George Orwell - 1984.doc - Microsoft Word'
}
```

Limits

What you cannot do in Python. :(

- Digital signing/verification.
- Form processing is twitchy.
  - RepotLab support is sketchy.
  - Utilities are clumsy and error prone.

Thumbnails

```
def timeout_alarm_handler(signum, frame):
       raise TimeOutAlarm

 try:
     signal.signal(signal.SIGALRM, timeout_alarm_handler)
     signal.alarm(CONVERSION_TIMEOUT_SECONDS)

     # convert -format pdf -[0] -thumbnail 175x -bordercolor white png:-
     converter = Popen(
         [
             '/usr/bin/convert', '-format', 'pdf', '-[0]',
             '-thumbnail', '175x', '-bordercolor', 'white', 'png:-',
         ],
         stdin=PIPE, stdout=sfile, stderr=efile,
     )

     shutil.copyfileobj(rfile, converter.stdin)
     converter.stdin.close()
     converter.communicate()
     signal.alarm(0)
 except TimeOutAlarm:
```

Hack: Text To PDF

Source URL: <http://djangosnippets.org/snippets/1778/>
A horrible ActiveState recipe that proves very useful.

```
pdf = pyText2Pdf(
    rfile, wfile,
    title=title,
    font=font,
    ptSize=font_size,
    tab=tab_width,
    cols=columns,
    do_ffs=do_ffs,
)
pdf.Convert()
```

- A single Python file, not even a module.
- Dumps text to a PDF file.

```
pdf = None
wfile.flush()
wfile.seek(0)
rfile.seek(0)
reader = PDFFileReader(wfile)
writer = PDFFileWriter(xfile)
writer.cloneDocumentFromReader(reader)
writer.addAttachment('source.txt', rfile.read())
writer.write(xfile)
```

Add the original text document as an attachment to the PDF version.

PyFDF

https://code.google.com/p/pyfpdf/wiki/Tutorial
https://github.com/reingart/pyfpdf

- Derived from the FPDF PHP class

```
import os
from fpdf import FPDF

pdf = FPDF()
pdf.add_page()
pdf.set_font('helvetica', '', 13.0)
pdf.set_xy(105.0, 8.0)
pdf.cell(ln=0, h=22.0, align='C', w=75.0, txt='Sample Invoice', border=0)
pdf.set_line_width(0.0)
...
```

Scrubbrush

```
self._cairo_path = self.context.server_defaults_manager.string_for_default(
    'PDFToCairoPath', value='/usr/bin/pdftocairo',
)
converter = None
converter_result = False
sfile = BLOBManager.ScratchFile(require_named_file=True, )
efile = BLOBManager.ScratchFile()
converter = Popen(
    [self._cairo_path, '-pdf', '-origpagesizes', '-', '-'], stdin=PIPE, stdout=sfile, stderr=efile,
)
try:
    signal.signal(signal.SIGALRM, timeout_alarm_handler)
    signal.alarm(CONVERSION_TIMEOUT_SECONDS)  # constant, for catching never ending scribs
    shutil.copyfileobj(rfile, converter.stdin)
    converter.stdin.close()
    converter.communicate()
    signal.alarm(0)
except TimeOutAlarm:
    converter_result = False
except Exception:
    converter_result = False
else:
    converter_result = True
finally:
    signal.alarm(0)
    BLOBManager.Close(efile)
    if converter.returncode is None:
        try:
            converter.terminate()
        except Exception:
            ...
if converter_result:
    BLOBManager.Close(rfile)
    sfile.seek(0)
    return sfile
```

PDF files, especially when moving between platforms, can have all manner of weird bugs – especially relating to fonts. The excellent libcairo can be used to scrub a PDF into maximum compatibility.

Beware: *font icons*, like check-marks etc… Especially web developers love that kind of junk – it looks cute – it **never** works consistently. Use images for icons and glyphs.

Scrubbrush

BEFORE:

name	type	encoding	emb	sub	uni	object	ID
CCQPLO+Arial	CID TrueType	Identity-H	yes	yes	yes	18	0
ArialMT	TrueType	WinAnsi	no	no	no	26	0
Arial-BoldMT	TrueType	WinAnsi	no	no	no	28	0
QVDGUR+MinionPro-Regular	CID Type 0C	Identity-H	yes	yes	yes	30	0
FZPPKI+MinionPro-Regular	CID Type 0C	Identity-H	yes	yes	yes	38	0
Helvetica-Bold	Type 1	Custom	no	no	no	52	0
Helvtica	Type 1	Custom	no	no	no	58	0
ZapfDeingbats	Type 1	ZapfDingbats	no	no	no	188	0

AFTER:

name	type	encoding	emb	sub	uni	object	ID
KOFYDJ+LiberationSans	TrueType	WinAnsi	yes	yes	yes	5	0
RXEXJO+LiberationSans-Bold	TrueType	WinAnsi	yes	yes	yes	6	0
IOKNUX+Arial	TrueType	WinAnsi	yes	yes	yes	7	0
AADISD+MinionPro-Regular	CID Type 0C	Identity-H	yes	yes	yes	10	0
AGFLBT+MinionPro-Regular	CID Type 0C	Identity-H	yes	yes	yes	11	0
THKLNC+NimbusSanL-Bold	Type 1	WinAnsi	yes	yes	yes	12	0
CIXCHU+NimbusSanL-Regu	Type 1	WinAnsi	yes	yes	yes	13	0
QAOVNF+Dingbats	Type 1	Builtin	yes	yes	yes	14	0

Cups

https://pypi.python.org/pypi/pycups/1.9.73

```
try:
    cups.setServer(IPP_SERVER_NAME)
    ipp_connection = cups.Connection()
    ipp_connection.printFile(
        print_queue,
        rfile.name,
        document.get_file_name(),
        {'media': 'Letter',
         'fit-to-page': str('yes'), })
    rfile.close()
    ipp_connection = None
except Exception as exc:
```

- License: GPLv2+