PDF secrets

hiding & revealing secrets in PDF documents

Ange Albertini

2014/09/05

ANGE ALBERTINI

reverse engineering

VISUAL DOCUMENTATIONS

@angealbertini

ange@corkami.com

http://www.corkami.com

Goal:
learn PDF *internals*

Application: hide/reveal content

UNCLASSIFIED

III. TRAFFIC CONTROL POINTS, BLOCKING POSITIONS, AND TRAINING

A. (U) Introduction

(U) This section examines TCPs, BPs, and training matters. It first discusses the difference between a TCP and a BP. Standing Operating Procedures (SOPs) for the various units involved regarding TCPs and BPs are assessed, and the Rhino Bus TTP is outlined. This is followed by a review of the training on TCPs, BPs, weapons, and Rules of Engagement (ROE) that the Soldiers manning BP 541 had received before 4 March 2005. The ROE that were in effect that night are explained. The section concludes with findings and recommendations.

B. (U) Traffic Control Points and Blocking Positions

(U) Task Force [redacted] had received missions to establish TCPs and blocking positions numerous times in the past. [redacted]

C. (U) Standing Operating Procedures in use on 4 March 2005

(U) SOPs are designed to serve as guidelines for specific operations and are not prescriptive in nature. They provide a baseline for acceptable operations from which commanders can derive principles and techniques and adapt them to their current mission. (Annexes 44C, 65C, 72C, 96C, 98C).

http://download.repubblica.it/pdf/rapportousacalipari.pdf

seen in its metadata: “EmailSubject (Another Redact Job For You)”

pocorgtfo05.pdf - Adobe Reader

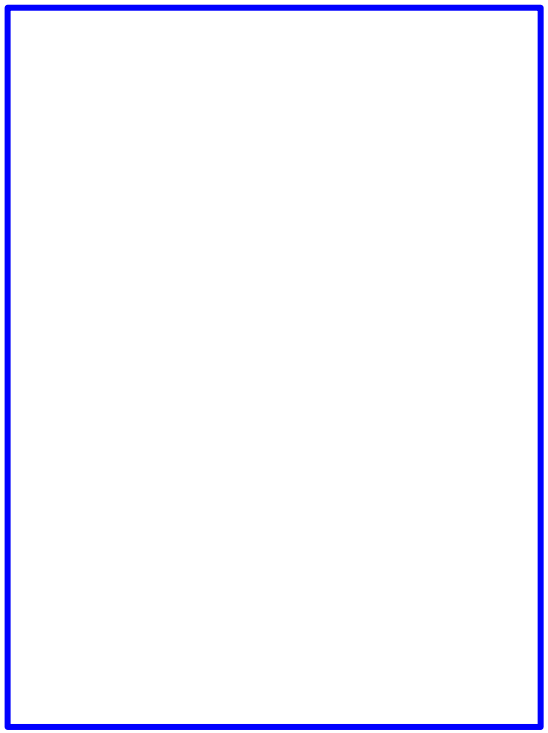
File Edit View Window Help

54 / 56 64.4% Tools Sign Comment

13 Ancestral Voices
Or, a vision in a nightmare.

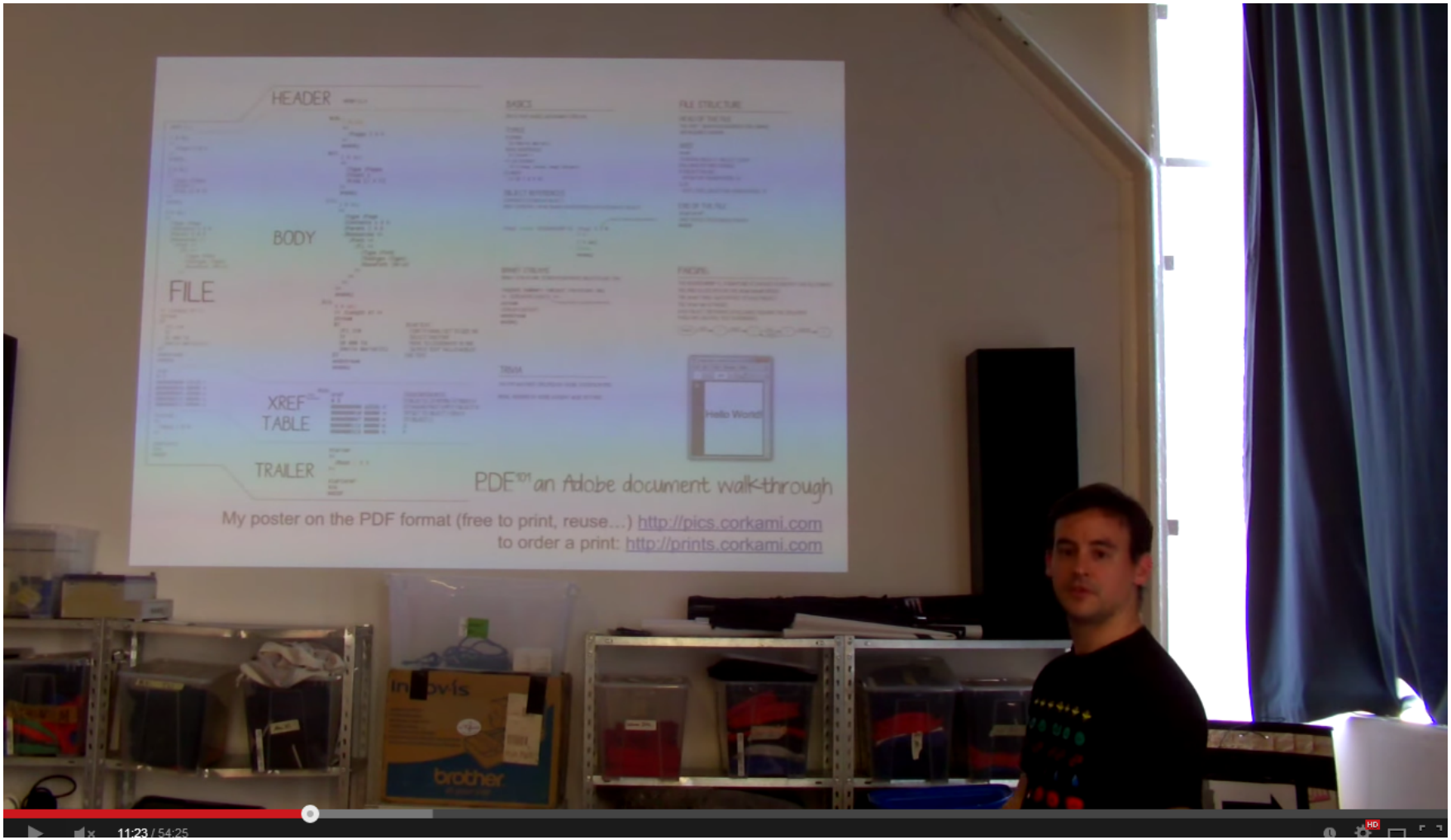
by Ben Nagy

This high-capacity, weaponized poem has been withheld from this international edition, as it may inspire new exploits and is thus a controlled export.[24]



[24]Look up Wassenaar Arrangement, *intrusion software*, *control lists*, and *controlled items*. If it helps develop, generate, or

This high-capacity, weaponized poem has been withheld from this international edition, as it may inspire new exploits and is thus a controlled export.[24]



https://www.youtube.com/watch?v=JQrBgVRgqtc

extra non-technical details

Preamble

this presentation has a lot of hands-on examples, that you can find at:

http://pdf.corkami.com

PDF 101

basics of the PDF file format

Part I / II

PDF 101 an Adobe document walk-through

FILE

```
%PDF-1.1

1 0 obj
<<
  /Pages 2 0 R
>>
endobj

2 0 obj
<<
  /Type /Pages
  /Count 1
  /Kids [3 0 R]
>>
endobj

3 0 obj
<<
  /Type /Page
  /Contents 4 0 R
  /Parent 2 0 R
  /Resources <<
    /Font <<
      /F1 <<
        /Type /Font
        /Subtype /Type1
        /BaseFont /Arial
      >>

<< /Length 47 >>
stream
BT
  /F1 110
  Tf
  10 400 Td
  (Hello World!)Tj
ET
endstream
endobj

xref
0 5
0000000000 65535 f
0000000010 00000 n
0000000047 00000 n
0000000111 00000 n
0000000313 00000 n

trailer
<<
  /Root 1 0 R
>>

startxref
416
%%EOF
```

HEADER

```
%PDF-1.1
```

BODY

```
010:
1 0 obj
<<
  /Pages 2 0 R
>>
endobj
047:
2 0 obj
<<
  /Type /Pages
  /Count 1
  /Kids [3 0 R]
>>
endobj
111:
3 0 obj
<<
  /Type /Page
  /Contents 4 0 R
  /Parent 2 0 R
  /Resources <<
    /Font <<
      /F1 <<
        /Type /Font
        /Subtype /Type1
        /BaseFont /Arial
      >>
    >>
  >>
>>
endobj
313:
4 0 obj
<< /Length 47 >>
stream
BT                      BEGIN TEXT
  /F1 110               FONT F1 (ARIAL) SET TO SIZE 110
  Tf                    SELECT THIS FONT
  10 400 Td             MOVE TO COORDINATE 10, 400
  (Hello World!)Tj      OUTPUT TEXT "HELLO WORLD!"
ET                      END TEXT
endstream
endobj
```

XREF TABLE

CROSS REFERENCE

```
416:
xref                    CROSS REFERENCES
0 5                     5 OBJECTS, STARTING AT INDEX 0
0000000000 65535 f      (STANDARD FIRST EMPTY OBJECT 0
0000000010 00000 n      OFFSET TO OBJECT 1, REV 0
0000000047 00000 n      TO OBJECT 2...
0000000111 00000 n      3...
0000000313 00000 n      4
```

TRAILER

```
trailer
<<
  /Root 1 0 R
>>

startxref
416
%%EOF
```

BASICS

PDF IS TEXT BASED, WITH BINARY STREAMS

TYPES

() STRING
  EX: `(Hello World!)`
/NAME (IDENTIFIERS)
  EX: `/Count 1`
<<>>: DICTIONARY
  EX: `<</key1 value1 /key2 value2>>`
[]: ARRAY
  EX: `[0 1 2 3 4]`

OBJECT REFERENCES

CONTENT IS STORED IN OBJECT
MOST CONTENT CAN BE INLINED OR REFERENCED IN A SEPARATE OBJECT

```
/Key1 value  IS EQUIVALENT TO  /Key1 3 0 R
                               [...]
                               3 0 obj
                               value
                               endobj
```

(3 0 R: <OBJECT NUMBER> <REVISION NUMBER> R)

BINARY STREAMS

BINARY STREAM ARE STORED IN SEPARATE OBJECTS LIKE THIS:

```
<object number> <object revision> obj
<< <STREAM METADATA> >>
stream
<STREAM CONTENT>
endstream
endobj
```

(<STREAM METADATA>: STREAM LENGTH, COMPRESSION PARAMETERS...)

TRIVIA

THE PDF WAS FIRST SPECIFIED BY ADOBE SYSTEMS IN 1993

INITIAL VERSIONS OF ADOBE ACROBAT WERE NOT FREE

FILE STRUCTURE

HEAD OF THE FILE

THE '%PDF-' SIGNATURE IDENTIFIES THE FORMAT
AND REQUIRED VERSION

XREF

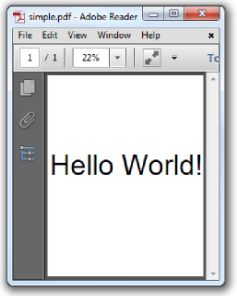
`xref`
<STARTING OBJECT> <OBJECT COUNT>
FOLLOWED BY XREF ENTRIES:
IF (OBJECT IN USE)
  <OFFSET:10> <GENERATION:5> `n`
ELSE
  <NEXT_FREE_OBJECT:10> <GENERATION:5> `f`

END OF THE FILE

`startxref`
<XREF OFFSET IN DECODED STREAM>
`%%EOF`

PARSING

THE HEADER `%PDF-1.?` SIGNATURE IS CHECKED TO IDENTIFY THE FILE FORMAT
THE XREF IS LOCATED VIA THE `startxref` OFFSET
THE `xref` TABLE GIVES OFFSET OF EACH OBJECT
THE `trailer` IS PARSED
EACH OBJECT REFERENCE IS FOLLOWED, BUILDING THE DOCUMENT
PAGES ARE CREATED, TEXT IS RENDERED

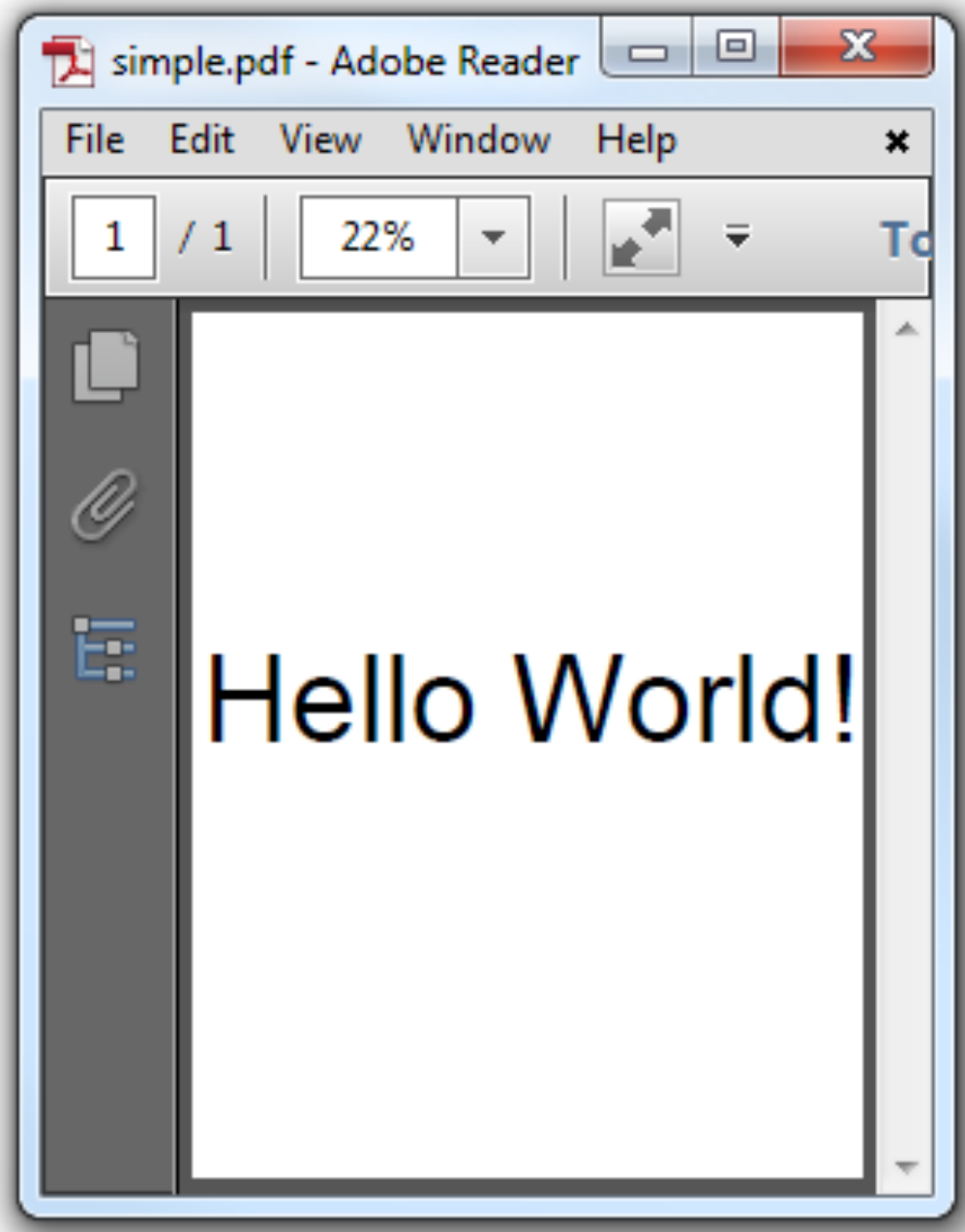


My poster on the PDF format (free to print, reuse…) http://pics.corkami.com
to order a print: http://prints.corkami.com

A simple example

helloworld.pdf

reminder: this is simplified, PDF is actually much more complex




```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R]  /Count 1  /Type /Pages >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Filter /FlateDecode /Length 57 >>
stream
xœs
áRPÐw3T044NUL²BÒ€„,i,‰BH
-á‘š““¯DLEž_”“¢¨©DLE’ÅåSUBÂENQNUL!0VT×
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000111 00000 n
0000000283 00000 n

trailer << /Root 1 0 R  /Size 5 >>

startxref
414
%%EOF
```

text

binary

text

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R]  /Count 1  /Type /Pages >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Filter /FlateDecode /Length 57 >>
stream
xœs
áRPÐw3T044NUL²BÒ€„,i,‰BH
-á‘š““¯DLEž_”“¢¨©DLE’ÅåSUBÂENQNUL!0VT×
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000111 00000 n
0000000283 00000 n

trailer << /Root 1 0 R  /Size 5 >>

startxref
414
%%EOF
```

A PDF file is

- text-based
  - white-space tolerant
- with binary streams

→ it can be explored with a decent text editor

if you need one, try Notepad++

http://notepad-plus-plus.org/

Recommended environment

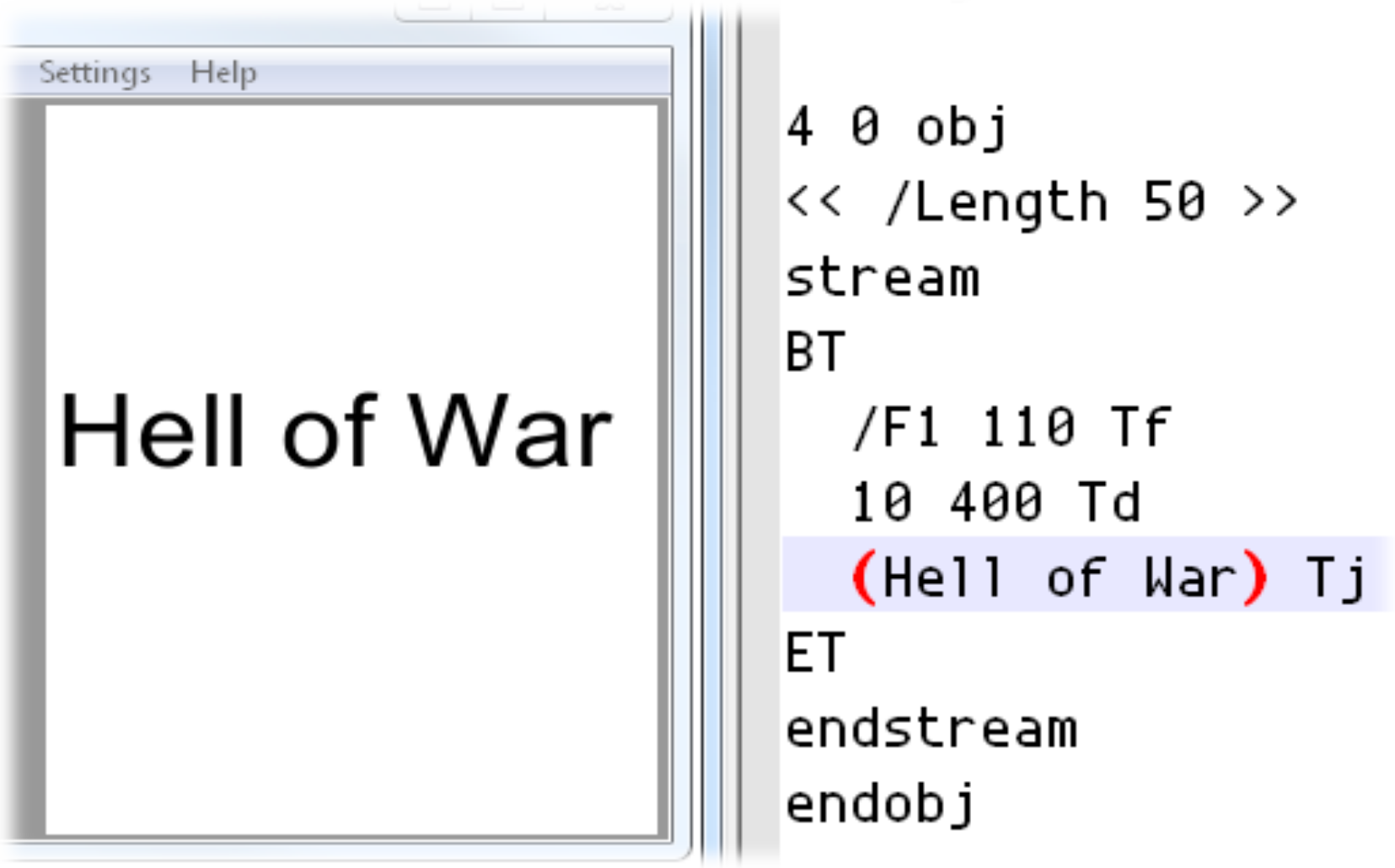
- text editor
- Evince/Sumatra
  - lightweight
  - updates on the fly

- a tool to decompress streams
  - (explanations later)

- check mistakes with `qpdf --check` or `pdfinfo`


```
4 0 obj
<< /Length 50 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!) Tj
ET
endstream
endobj
```

Update content, save...



...and you see the result straight away.

A PDF structure

1. header
   - signature
2. body
   - objects
3. cross-reference table
4. trailer
   - cross-reference table
   - trailer dictionary
   - xref pointer
   - end of file signature

Signature

1. PDF signature
   - `%PDF-1.0 - %PDF-1.7`
2. charset identifier
   - not required
   - tells tools it's not ASCII
   - 4 non-ASCII chars in a comment

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R]  /Count 1  /Type /Pages >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Filter /FlateDecode /Length 57 >>
stream
xœs
áRPÐw3T044NUL²BÒ€„,i,‰BH
-á‘š““¯DLEž_”“¢¨©DLE’ÅåSUBÂENQNUL!0VT×
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000111 00000 n
0000000283 00000 n

trailer << /Root 1 0 R  /Size 5 >>

startxref
414
%%EOF
```

Body

made of objects

- <number> <generation> **obj**
  <content>
  endobj

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R]  /Count 1  /Type /Pages >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Filter /FlateDecode /Length 57 >>
stream
xœs
áRPÐw3T044NUL²BÒ€„,i,‰BH
-á‘š““¯DLEž_”“¢¨©DLE’ÅåSUBÂENQNUL!0VT×
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000111 00000 n
0000000283 00000 n

trailer << /Root 1 0 R  /Size 5 >>

startxref
414
%%EOF
```

Xref

- table
- offsets of each object

```
xref
0 5                     5 objects, starting at 0
0000000000 65535 f      obj #0: always null
0000000016 00000 n      obj #1: offset 16
0000000051 00000 n      obj #2: offset 51
0000000111 00000 n      ...
0000000283 00000 n
```

- each line = 20 chars
  - space before CR

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R]  /Count 1  /Type /Pages >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Filter /FlateDecode /Length 57 >>
stream
xœs
áRPÐw3T044NUL²BÒ€„,i,‰BH
-á‘š““¯DLEž_”“¢¨©DLE’ÅåSUBÂENQNUL!0VT×
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000111 00000 n
0000000283 00000 n

trailer << /Root 1 0 R  /Size 5 >>

startxref
414
%%EOF
```

Trailer 1/2

- structure
  a. “`trailer`”
  b. dictionary
     (like most objects)

- defines the “root” object
  - `/Size` = #(xref elements)

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R]  /Count 1  /Type /Pages >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Filter /FlateDecode /Length 57 >>
stream
xœs
áRPÐw3T044NUL²BÒ€„i,%BH
-á‘š““¯DLEž_”“¢¨©DLE’ÅåSUBÂENQNUL!0VT×
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000111 00000 n
0000000283 00000 n

trailer << /Root 1 0 R  /Size 5 >>

startxref
414
%%EOF
```

Trailer 2/2

1. pointer to xref
   a. “startxref”
   b. offset to xref
      - (decimal)
2. End Of File marker
   a. %%EOF

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R]  /Count 1  /Type /Pages >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Filter /FlateDecode /Length 57 >>
stream
xœs
áRPÐw3T044NUL²BÒ€„i,%BH
-á‘š““¯DLEž_”“¢¨©DLE’ÅåSUBÂENQNUL!0VT×
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000111 00000 n
0000000283 00000 n

trailer << /Root 1 0 R  /Size 5 >>

startxref
414
%%EOF
```

Basic types

names, strings, dictionaries...

Literals

- `%comment until line return`
- `(string)`
- `<hex>`

- some others, less-used types
  (PDF is *quite* f*cked up)

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R] /Type /Pages /Count 1 >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Length 53 >>
stream
BT
  /F1 110
  Tf
  10 400 Td
  (Hello World!) Tj
ET
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000109 00000 n
0000000281 00000 n

trailer << /Root 1 0 R /Size 5 >>

startxref
384
%%EOF
```

equivalent files

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R] /Type /Pages /Count 1 >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Length 75 >>
stream
BT
  /F1 110
  Tf
  10 400 Td
  <48 65 6C 6C 6F 20 57 6F 72 6C 64 21> Tj
ET
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000109 00000 n
0000000281 00000 n

trailer << /Root 1 0 R /Size 5 >>

startxref
407
%%EOF
```

Object reference

points

- `<object> <generation>` **R**

to

- the actual contents of the object

some object CAN'T be inlined

`<generation>` is *very rarely* non-null

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R] /Count 1 /Type /Pages >>
endobj

3 0 obj
<< /Parent 2 0 R /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Filter /FlateDecode /Length 57 >>
stream
xœs
áRPÐw3T044NUL²BÒ€„,i,‰BH
-á‘š““¯DLEž_”“¢¨©DLE’ÅåSUBÂENQNUL!0VT×
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000111 00000 n
0000000283 00000 n

trailer << /Root 1 0 R /Size 5 >>

startxref
414
%%EOF
```

Object reference - example 1

```
57
...
```

```
354 0 R
...

354 0 obj
57
endobj
```

2 equivalent examples via object reference

Object reference syntax

it's odd (PostScript), but critical to understand

- `3 0 1` ⇒ 3 elements (3 numbers):
  a. `3`
  b. `0`
  c. `1`
- `3 0 R` ⇒ 1 element:
  a. reference to "`3 0`"
     - object `3`
     - generation `0`

Other PDF syntax rules follow common-sense

Name objects

- “reserved keywords”
  - like symbols in Ruby
- starts with */*
  - `/Pages` , `/Kids` …

- case sensitive
  - CamelCase by default
  - undefined names are ignored

⇒`/pages != /Pages`

(useful to disable tags)

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R]  /Count 1  /Type /Pages >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Filter /FlateDecode /Length 57 >>
stream
xœs
áRPÐw3T044NUL²BÒ€„i,‰BH
-á‘š“““¯DLEž_”“¢¨©DLE’ÅåSUBÂENQNUL!0VT×
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000111 00000 n
0000000283 00000 n

trailer << /Root 1 0 R  /Size 5 >>

startxref
414
%%EOF
```

Array

Syntax

- `[ <values>* ]`

Examples:

- `[3 0 R]` = 1 value
  a. “3 0 R”
- `[0 0 612 792]` = 4 values
  a. 0
  b. 0
  c. 612
  d. 792

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R]  /Count 1  /Type /Pages >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Filter /FlateDecode /Length 57 >>
stream
xœs
áRPÐw3T044NUL²BÒ€„,i,‰BH
-á‘š“““¯DLEž_”“¢¨©DLE’ÅåSUBÂENQNUL!0VT×
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000111 00000 n
0000000283 00000 n

trailer << /Root 1 0 R  /Size 5 >>

startxref
414
%%EOF
```

Dictionaries

Syntax:

- **<<** [<name> <value>]* **>>**

Object 1 sets:

1. */Pages* to “2 0 R”

Object 2 sets:

1. */Kids* to “[3 0 R]”
2. */Count* to “1”
3. */Type* to */Pages*

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R]  /Count 1  /Type /Pages >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Filter /FlateDecode /Length 57 >>
stream
xœs
áRPÐw3T044NUL²BÒ€„,i,‰BH
-á‘š“““¯DLEž_”“¢¨©DLE’ÅåSUBÂENQNUL!0VT×
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000111 00000 n
0000000283 00000 n

trailer << /Root 1 0 R  /Size 5 >>

startxref
414
%%EOF
```

Object reference - example 2

```
/Pages 2 0 R
```

is “equivalent” to

```
/Pages <<
   /Kids [3 0 R]
   /Count 1
   /Type /Pages
>>
```

```
1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R] /Count 1 /Type /Pages >>
endobj
```

and then ”`3 0 R`“ is a further reference…

Binary streams

parameters, filters...

Streams

syntax:

1. usual object declaration
2. parameters dictionary
3. `stream`
   `+ return character`
4. stream data
5. `endstream`
   `+ return character`
6. usual endobj

stream data is not interpreted

(at object level)

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R]  /Count 1  /Type /Pages >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Filter /FlateDecode /Length 57 >>
stream
xœs
áRPÐw3T044NUL²BÒ€„,i,‰BH
-á‘š““¯DLEž_”“¢¨©DLE’ÅåSUBÂENQNUL!0VT×
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000111 00000 n
0000000283 00000 n

trailer << /Root 1 0 R  /Size 5 >>

startxref
414
%%EOF
```

Example

object 4

- stream parameters
  - `/Filter = /FlateDecode`
  - `/Length = 57`
- stream content (binary)

  xœsáRPÐw3T044²BÒ€„¡□,‰□□BH

  □-á‘š“”¯ž_”“¢¨©’ÅåÂ !0×

```
4 0 obj
<< /Filter /FlateDecode /Length 57 >>
stream
xœs
áRPÐw3T044NUL²BÒ€„¡,‰BH
-á‘š“”“¯DLEž_”“¢¨©DLE’ÅåSUBÂENQNUL!0VT×
endstream
endobj
```

Binary streams

- can be stored with different encodings
  - `/Filter`
  - encodings can be cascaded
- content is decoded
  - after each filter

only the final data matters

Streams don’t enforce encodings

as long as the result is correct
once decoded by the filters

```
<< /Length 53 >>

stream
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!) Tj
ET
endstream
```

```
<< /Length 57
   /Filter /FlateDecode
>>

stream
xœs
áRPÐw3T044 ²BÒ€„¡‚‰BH
-á‘š“““-ž_”“¢¨©’ÅåÂ !0×
endstream
```

these 2 streams are equivalent, just using a different encoding
(DEFLATE = ZIP compression)

```
<< /Length 170
   /Filter [
      /ASCIIHexDecode
      /FlateDecode] >>
stream
78 9C 73 0A E1 52 50 D0 77 33 54 30 34
34 00 B2 42 D2 80 84 A1 81 82 89 81 81
42 48 0A 90 AD E1 91 9A 93 93 AF 10 9E
5F 94 93 A2 A8 A9 10 92 C5 E5 1A C2 05
00 21 30 0B D7
endstream
```

```
<< /Length 57
   /Filter /FlateDecode
>>

stream
xœs
áRPÐw3T044 ²BÒ€„¡‚‰BH
-á‘š““-ž_”“¢¨©’ÅåÂ !0×
endstream
```

/ASCIIHexDecode will decode ASCII Hex to binary, then Deflating will decompress the result

Main filters

- `<none>`: direct raw binary in the file
- `/FlateDecode` : ZIP's deflate decompression

  → smaller
- `/ASCIIHexDecode:` turns hex into binary
  - `41 0A` ⇒ "`A\n`"

  → easy text editing (but binary is very common)

  `mutool` has a specific option for that

Other filters

Images

- `/DCTDecode` to store JPEG **files** directly
  - not just the data, even the header!
- JPEG2000, Fax

Encryption

- Crypt
  - RC4 or AES

Let’s put it all together

how is the file actually parsed?

Parsing 1/7

1. Signature is checked

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R] /Type /Pages /Count 1 >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Length 53 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!) Tj
ET
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000109 00000 n
0000000281 00000 n

trailer << /Root 1 0 R /Size 5 >>

startxref
384
%%EOF
```

Parsing 2/7

2. %%EOF is located

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R] /Type /Pages /Count 1 >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Length 53 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!) Tj
ET
endstream
endobj

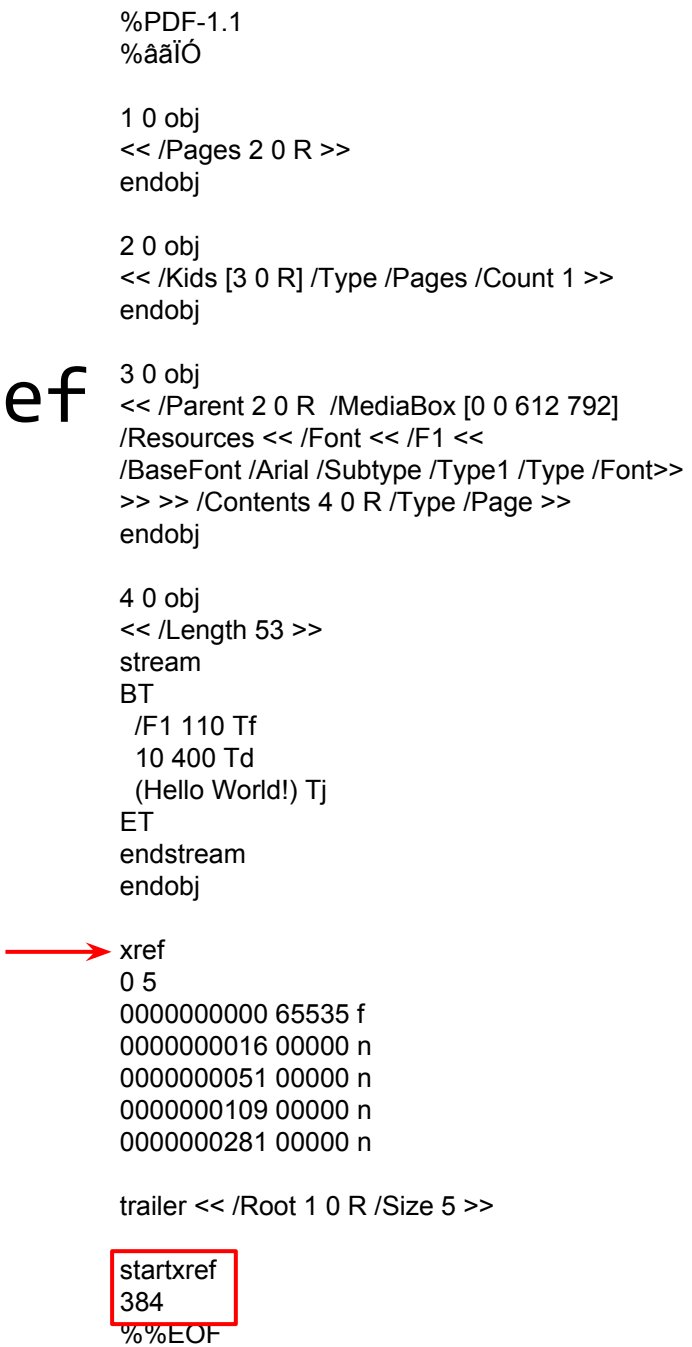
xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000109 00000 n
0000000281 00000 n

trailer << /Root 1 0 R /Size 5 >>

startxref
384
%%EOF
```

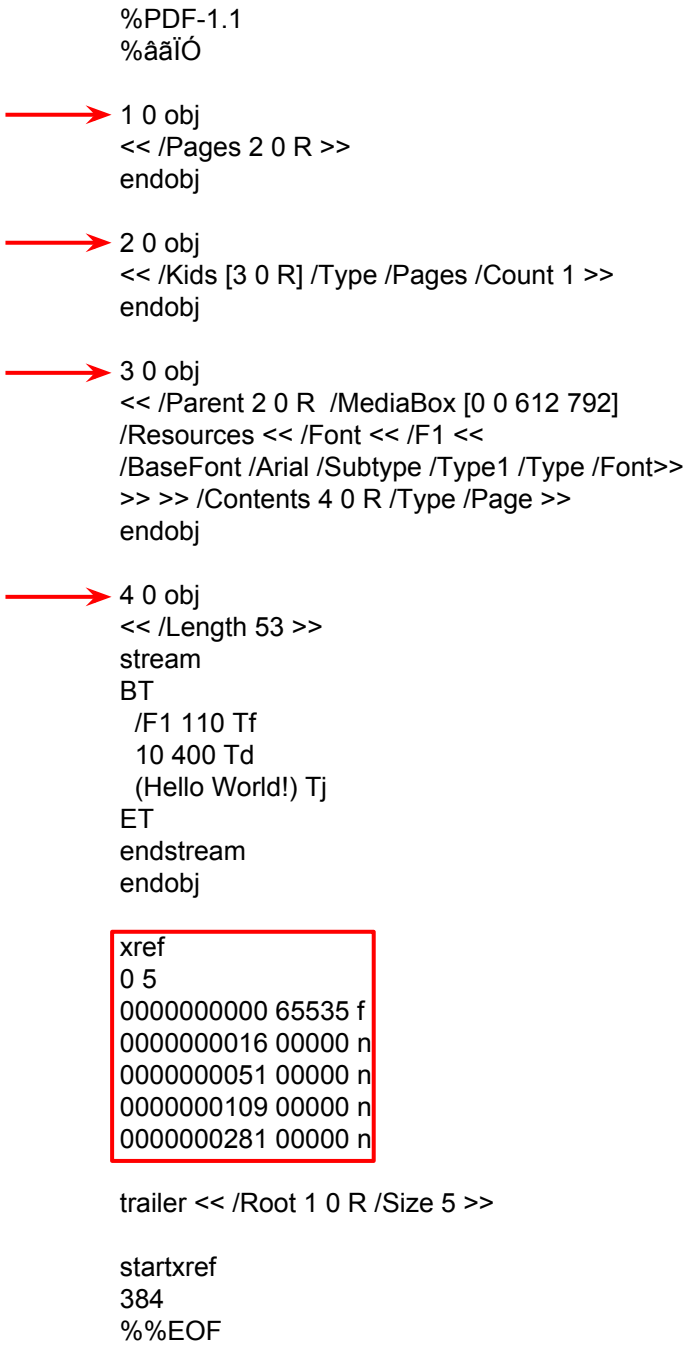
Parsing 3/7

3. `xref` is located via `startxref`



Parsing 4/7

4. `xref` gives the offsets of each objects



Parsing 5/7

5. `trailer` is parsed
   → gives `/Root` object

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R] /Type /Pages /Count 1 >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Length 53 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!) Tj
ET
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000109 00000 n
0000000281 00000 n

trailer << /Root 1 0 R /Size 5 >>

startxref
384
%%EOF
```

Parsing 6/7

6. objects are parsed
   a. `/Root` object contains `/Pages`
   b. `/Pages` contains page array
      - `/Kids`
   c. each `/Page` has:
      - size: `/MediaBox`
      - `/Contents`
        - as stream object
      - `/Resources`
        - defines the `/Font` dictionary

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R] /Type /Pages /Count 1 >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Length 53 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!) Tj
ET
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000109 00000 n
0000000281 00000 n

trailer << /Root 1 0 R /Size 5 >>

startxref
384
%%EOF
```

Parsing 7/7

7. the page is rendered
   a. `BT` BeginText
   b. `<name> <size> Tf` select font
   c. `<x> <y> Td` move cursor
   d. `<string> Tj` display string
   e. `ET` EndText


```
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!) Tj
ET
```

```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R] /Type /Pages /Count 1 >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Length 53 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!) Tj
ET
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000051 00000 n
0000000109 00000 n
0000000281 00000 n

trailer << /Root 1 0 R /Size 5 >>

startxref
384
%%EOF
```

In practice

- that was the ‘strict’ minimum
- a typical PDF embeds more information
  - fonts
  - fonts encoding
  - metadata
  - …

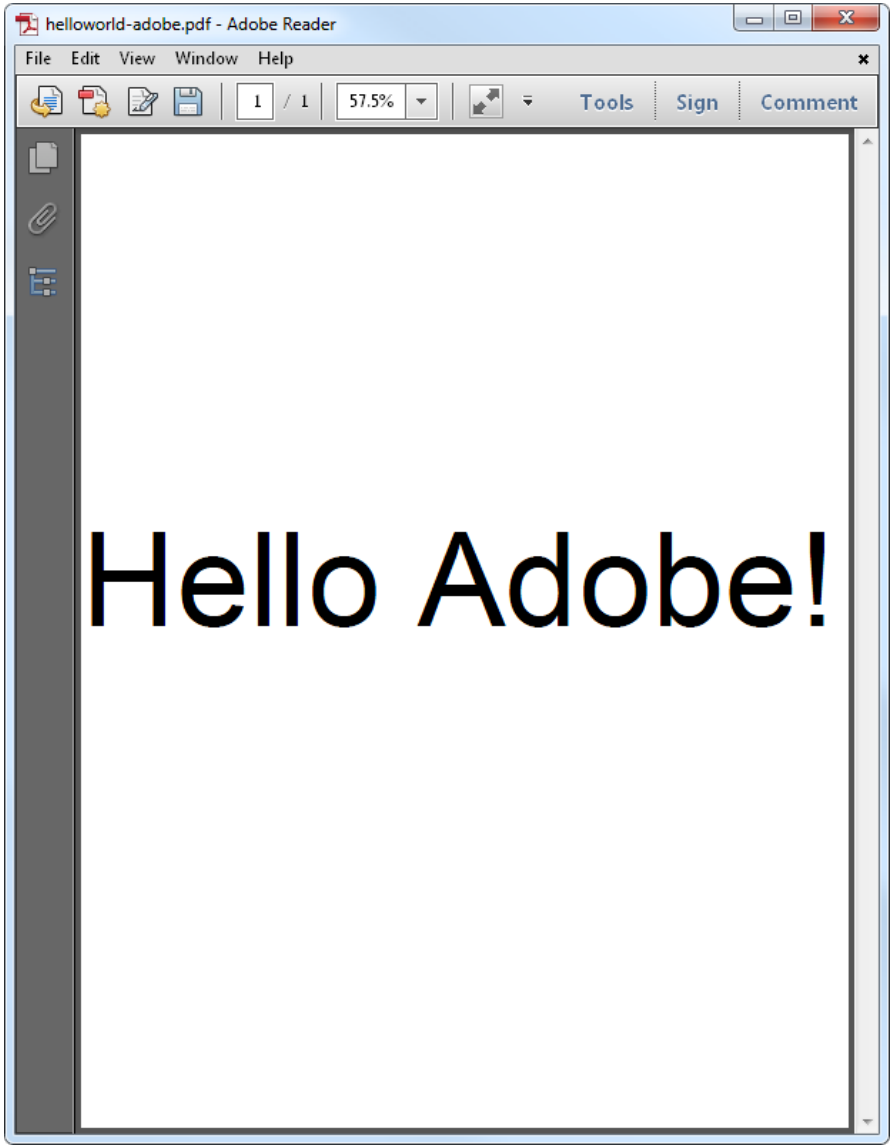
a generated *Hello World* typically weights >5 Kb

In practice - in the malware world

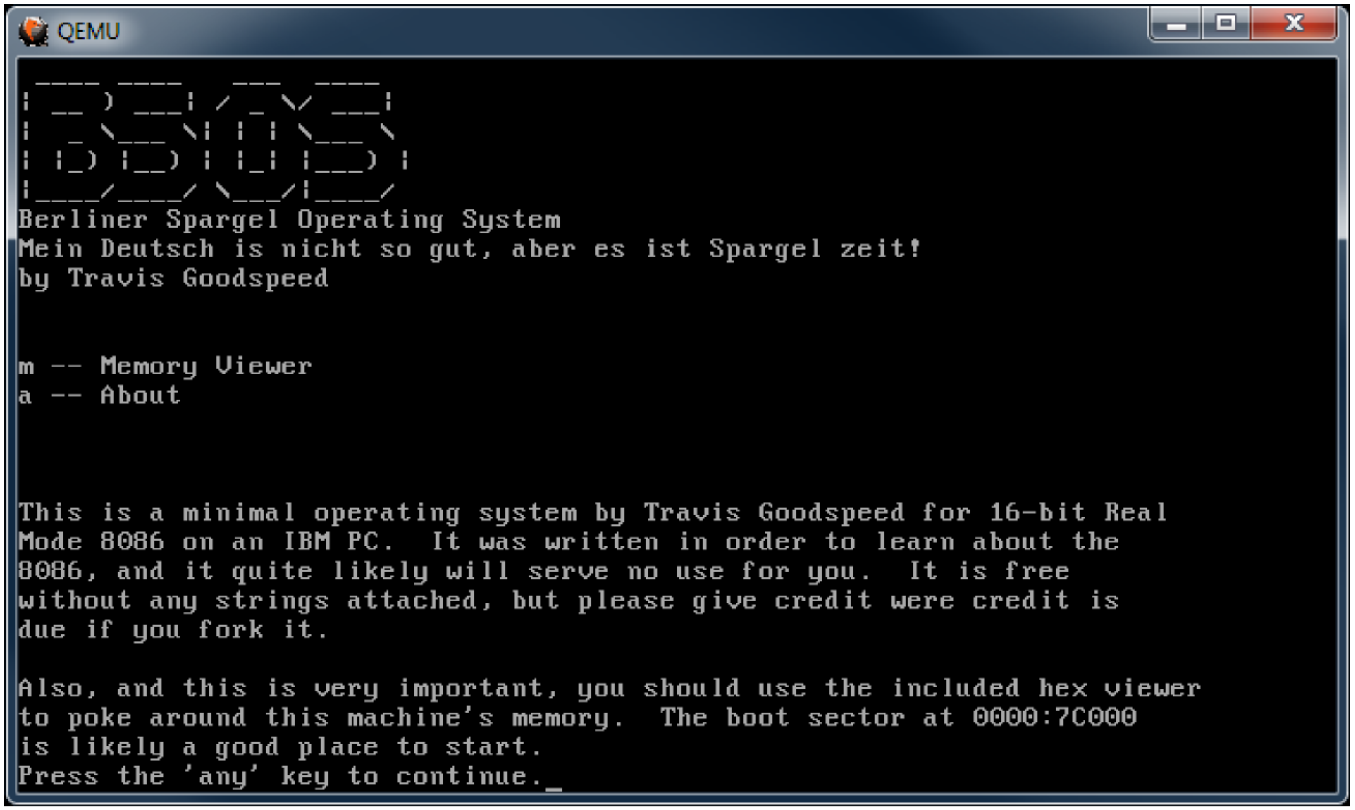
- most readers accept malformed files
  - many elements missing
    - EOF, startxref, xref, /Length, endobj, endstream
    - /MediaBox /Font
- each reader has its own weirdness
  - see my “Schizophrens” talks and PoCs

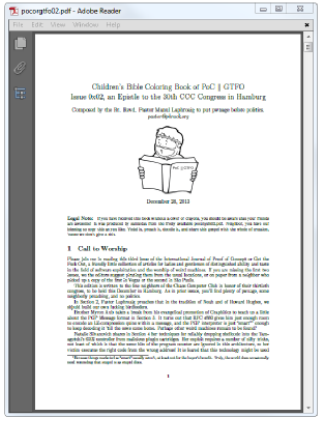
- so much for the so-called “standard”

```
%PDF-\01 0 obj<</Kids
[<</Parent 1 0 R/Contents
[2 0 R]>>]
/Resources<<>>>>2 0
obj<<>>stream\n
BT/F1 105 Tf 0 400 Td
(Hello Adobe!)Tj ET
endstream\n
endobj\n
trailer<</Root<</Pages 1
0 R>>>>
```



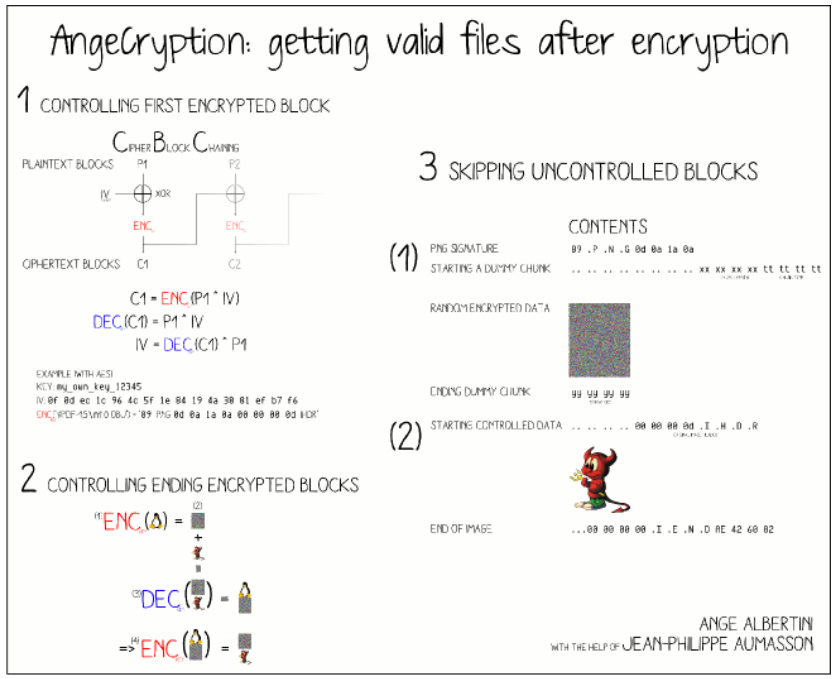
a “Hello World” for Adobe, in 179 bytes

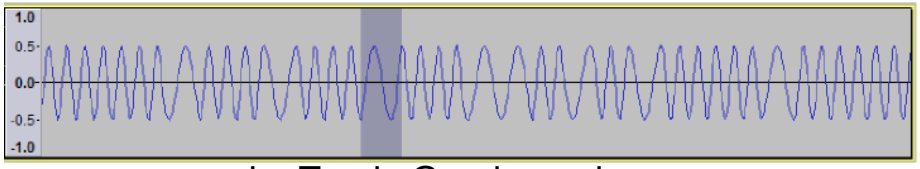




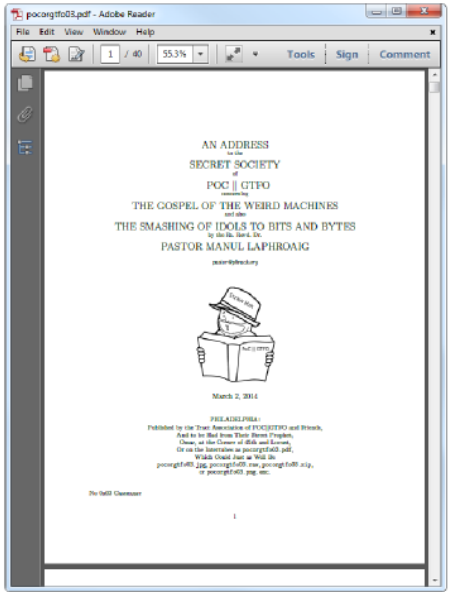
```
Archive:  pocorgtfo02.pdf
warning [pocorgtfo02.pdf]:  8016414 extra bytes at beginning or within zipfile
  (attempting to process anyway)
  Length      EAs    ACLs    Date    Time    Name
 --------     ---    ----    ----    ----    ----
      852       0       0  12/06/13 16:25   README.txt
     6794       0       0  12/06/13 16:25   coda.txt
    20164       0       0  12/06/13 16:25   feeling.txt
    12618       0       0  12/06/13 16:25   harrison.txt
        0       0       0  12/06/13 16:25   pgpquine/
      275       0       0  12/06/13 16:25   pgpquine/Makefile
     1006       0       0  12/06/13 16:25   pgpquine/inflate.c
     5323       0       0  12/06/13 16:25   pgpquine/quine.c
   203706       0       0  12/06/13 16:25   rfc4880.txt
  2046109       0       0  12/06/13 16:25   tamagotchi.zip
    15565       0       0  12/06/13 16:25   thewub.txt
   278598       0       0  08/05/13 13:06   pocorgtfo00.pdf
  3790438       0       0  10/13/13 02:47   pocorgtfo01.pdf
 --------     -----   -----                   -------
  6381448       0       0                   13 files
```

PoC||GTFO 0x2: MBR || PDF || ZIP



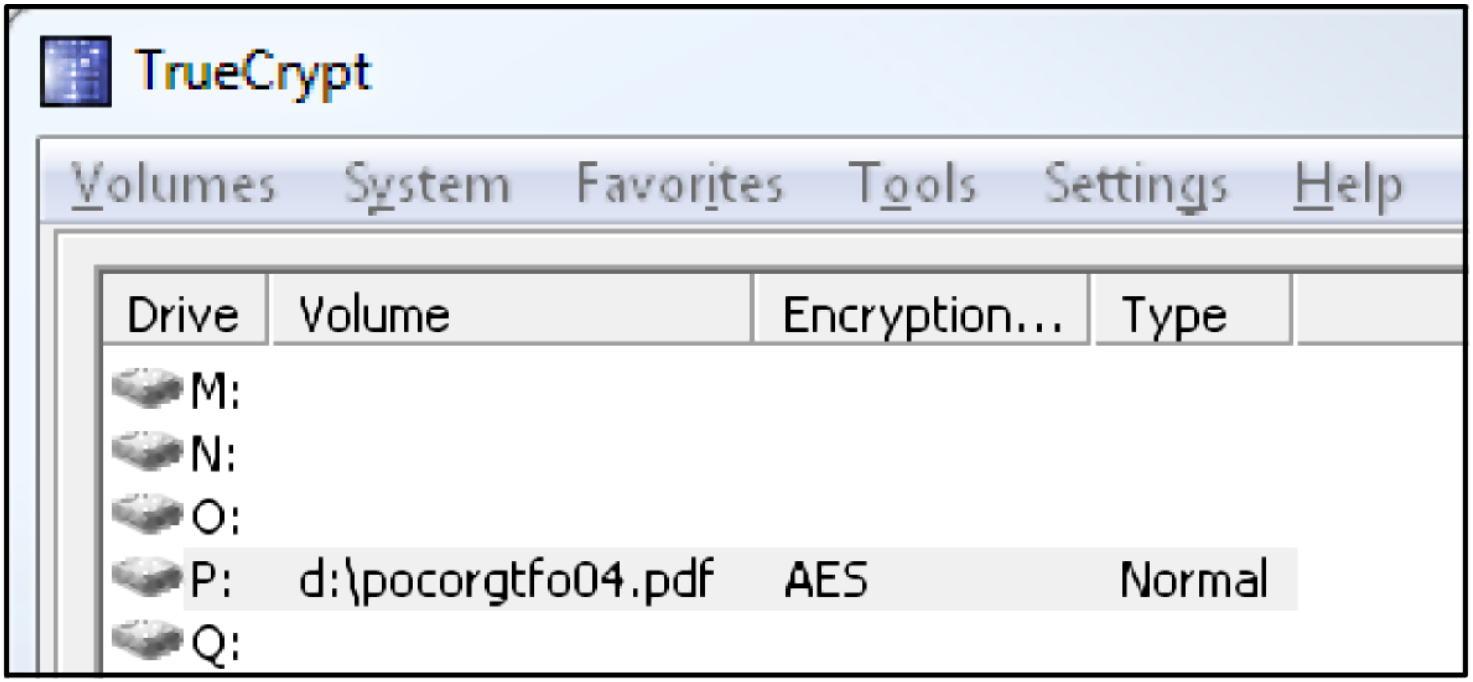


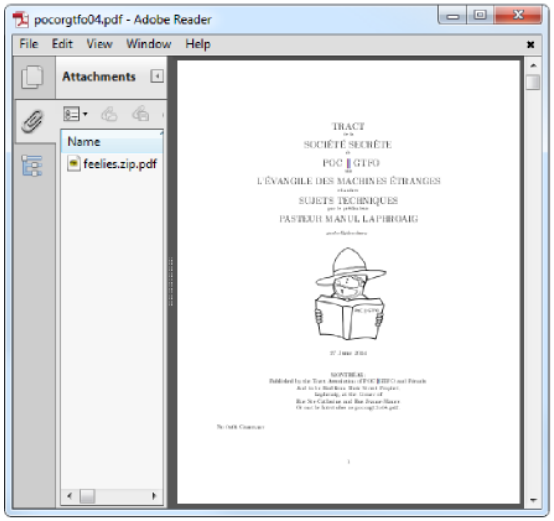
by Travis Goodspeed




```
Archive:  pocorgtfo03.pdf
warning [pocorgtfo03.pdf]:  12224072 extra bytes at beginning or within zipfile
  (attempting to process anyway)
  Length      EAs    ACLs    Date    Time    Name
 --------     ---    ----    ----    ----    ----
     2561       0       0  02/10/14 06:23   alexander.txt
     7848       0       0  02/08/14 20:20   bochs-2.6.2.patch
     6135       0       0  02/08/14 20:21   bochs-20140203.patch
     7248       0       0  02/09/14 08:35   defusing.zip
     4830       0       0  12/01/13 15:48   despair.txt
    14892       0       0  11/27/13 19:03   lasta.txt
    26325       0       0  02/07/14 21:06   lastq.txt
   473449       0       0  02/07/14 21:06   netwatch-337f8b1.tar.gz
   131930       0       0  02/24/14 20:32   nokiacipher.png
    14645       0       0  02/17/14 18:52   packed
     2129       0       0  02/07/14 21:06   saucers.txt
     3144       0       0  02/07/14 21:06   tamadec.txt
     6227       0       0  02/07/14 21:06   tetranglix.tar.bz2
 14109425       0       0  02/07/14 21:06   pocorgtfo02.pdf
      322       0       0  03/03/14 01:28   pocorgtfo03-encrypt.py
 --------     -----   -----                   -------
 14811110       0       0                   15 files
```

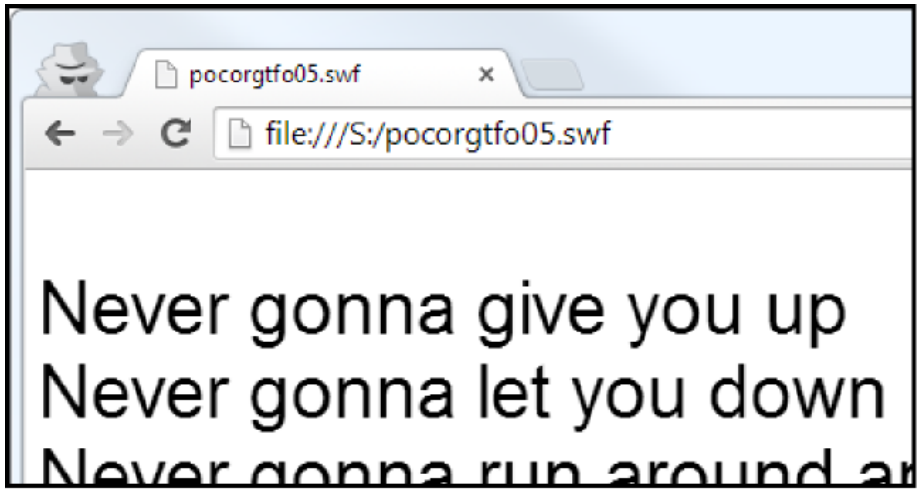
PoC||GTFO 0x3: JPG || AFSK || AES(PNG) || PDF || ZIP



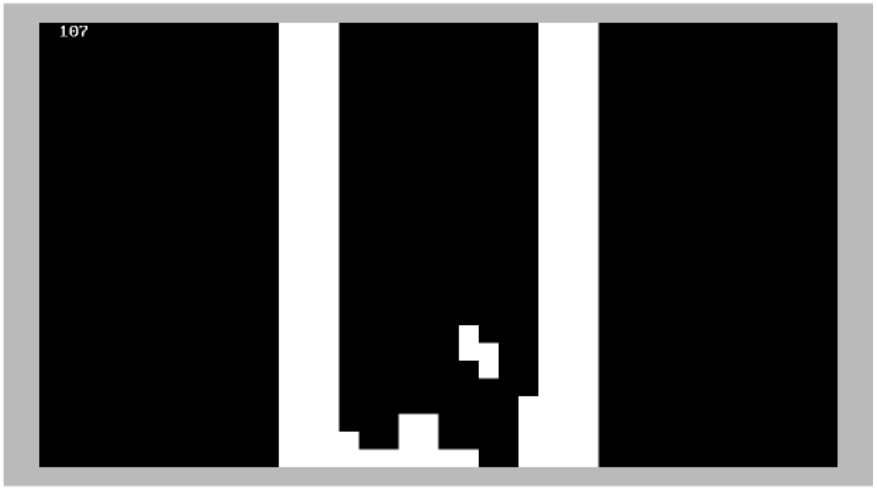


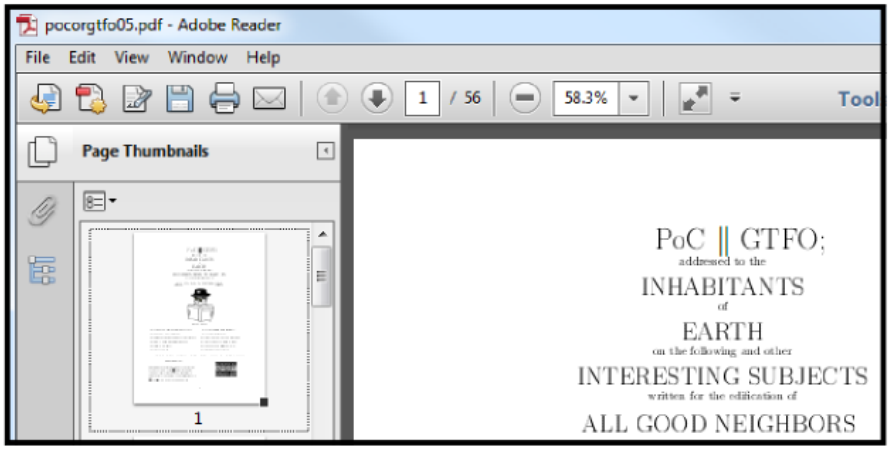

```
Archive:  pocorgtfo04.pdf
warning [pocorgtfo04.pdf]:  798586 extra bytes at beginning or within zipfile
  (attempting to process anyway)
error [pocorgtfo04.pdf]:  reported length of central directory is
  -798586 bytes too long (Atari STZip zipfile?  J.H.Holm ZIPSPLIT 1.1
  zipfile?).  Compensating...
  Length      EAs    ACLs    Date    Time    Name
 --------     ---    ----    ----    ----    ----
        0       0       0  06/24/14 18:56   bin2png/
     5010       0       0  06/24/14 18:56   bin2png/bin2png.py
    18025       0       0  06/24/14 18:56   bin2png/LICENSE
     1141       0       0  06/24/14 18:56   bin2png/README.md
   140413       0       0  06/24/14 18:56   darfsteller.txt
     2841       0       0  06/24/14 18:56   gods.txt
        0       0       0  06/24/14 18:56   lenticrypt/
    36445       0       0  06/24/14 18:56   lenticrypt/lenticrypt.py
    18025       0       0  06/24/14 18:56   lenticrypt/LICENSE
      776       0       0  06/24/14 18:56   lenticrypt/README.md
     2709       0       0  06/24/14 18:56   lenticrypt/test.py
  3111965       0       0  06/24/14 18:56   pocorgtfo.png
    25986       0       0  06/24/14 18:56   theveldt.txt
   239224       0       0  06/24/14 18:56   tsb-20140401.zip
 26750864       0       0  06/24/14 18:56   pocorgtfo03.pdf
 --------     -----   -----                   -------
 30353424       0       0                   15 files
```

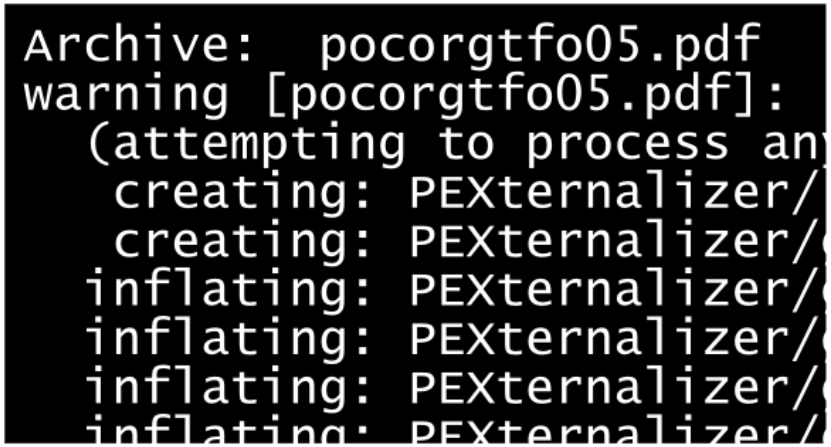
PoC||GTFO 0x4: TrueCrypt || PDF || ZIP



by Alex Inführ







PoC||GTFO 0x5: Flash || ISO || PDF || ZIP

Reminders on syntax

basic ones

% comment until line return

<hex string>
(standard string)

Equivalent examples:
(Hello World!)
<48 65 6c 6c 20 57 6f 72 64 21>

dictionary

<< [/name value]* >>

<< /Size 637 >> sets /Size to 637

Ex:

<< /Creator (Ange Albertini) >>

sets /Creator to "Ange Albertini"

Array

[ ]: Array

[0 0 612 792] : array of 4 elements

binary streams

absolutely ***anything*** between

stream

endstream

inside a dedicated object

with stream encoding parameters

in the object's dictionary

backward syntaxes

Because PDF encapsulates Postscript

References

1 0 R : refers to object 1 generation 0

refers to what's between

1 0 obj

endobj

Example:

[1 0 R] is an array of one element

which is one reference to object "1 0"

Page contents

inside a binary stream

- /F1 110 Tf: uses text font F1 with size 110
- 10 400 Td: puts cursor at x=10 y=400
- (Hello World) Tj : prints Hello World

Walkthrough

```
%PDF-1.1
%âãÏÓ

1 0 obj <<
  /Pages 2 0 R
>>
endobj

2 0 obj <<
  /Type /Pages
  /Count 1
  /Kids [3 0 R]
>>
endobj

3 0 obj <<
  /Type /Page
  /Parent 2 0 R
  /MediaBox [0 0 612 792]
  /Resources <<
    /Font <<
      /F1 <<
        /Type /Font
        /Subtype /Type1
        /BaseFont /Arial
      >>
    >>
  >>
  /Contents 4 0 R
>>
endobj

4 0 obj
<< /Length 51 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!) Tj
ET
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000053 00000 n
0000000117 00000 n
0000000345 00000 n

trailer <<
  /Root 1 0 R
  /Size 5
>>

startxref
446
%%EOF
```

Page's /Resources

```
/Resources
<<
  /XObject <</Im0 5 0 R>>
  ..
>>
```

Image object:

```
5 0 obj
<<
  /Type /XObject
  /Subtype /Image
  /Width <width>
  /Height <height>
  /BitsPerComponent 8
  /ColorSpace /DeviceRGB
  /Filter [
    /ASCIIHexDecode
    /DCTDecode % JPEG only
  ]
>>
stream
<IMAGE DATA>
endstream
endobj
```

Page's /Contents object stream:

```
q
<width> 0 0 <height> 0 0 cm
/Im0 Do
Q
```

Using an image in a PDF

Images = independant objects

They can be dumped by trivial parsing

Conclusion

we've covered the basics of:

- file structure
- objects relation
- file parsing
- page rendering

→ enough to play with PDF internals!

Hiding/revealing elements

Part II / II

text can be copied
images can be extracted

the “Select All” trick often works, but not always

even if “Select All” does *not* work, secrets *may* still be recovered

Reader-specific hiding

via trailer parsing schizophrenia

- Decoy + real PDF documents
  - decoy viewable with Adobe, Evince, Chrome
    extractable with pdftotext
  - real PDF viewable via Sumatra

⇒ avoid automated extraction

/!\ images = trivial to dump

Hiding external data in PDFs

- insert bogus object containing anything
  a. append or prepend:

     [%PDF-1.4] ⇐ if prepend

     ```
     999 0 obj
     stream
     <data>
     endstream
     ```
  b. adjust XREF

Elegant use: bundle sources with paper

hiding/revealing parts of the PDF document

from this point on:
not hiding data in a PDF file (stego)
nothing reader-specific (schizo)

Isn’t copy/paste enough?

- why not editing the file itself ?
  and restoring the secrets perfectly?

want to hide something?

- create your own methods!

Easy PDF editing

1. decompress streams
   - PDFTk , qpdf
   - optional: use ASCIIHex to get an ASCII-only file
2. open in text editor
3. view results via Sumatra

overwrite, or comment (don’t delete)

⇒ no offset to adjust

```
D:\>pdftk "PDF Secrets.pdf" output uncompressed.pdf uncompress
```

```
D:\>qpdf  --qdf "PDF Secrets.pdf" uncompressed.pdf
```

Reminder

technically speaking, a PDF page is:

1. a stream object
2. as the `/Contents` of a `/Type /Page` object
3. in the `/Kids` array of a `/Type /Pages` object
4. as the value of `/Pages` in root object
5. as the value of `/Root` in the trailer

and a text on the page is a simple `(`*`string`*`) Tj`

Remove a page ?

easy hiding

1. remove reference from `/Kids`
2. write it back later

```
obj
15776
endobj
1
0
obj
<<
/Type
/Pages
/Kids
[
6
0
R
14
0
R
21
0
R
]
/Count
3
>>
endobj
xref
0 41
0000000002 65535 f
0000117809 00000 n
0000000003 00000 f
0000000000 00000 f
0000000016 00000 n
0000000160 00000 n
0000000207 00000 n
0000000373 00000 n
0000083202 00000 n
0000000730 00000 n
```

Ln : 1697 | UNIX | ANSI | OVR

my public prezo

this slide should deniably removed

private text

and private image:

public slide

public text

locate the /Kids array

```
obj
15776
endobj
1
0
obj
<<
/Type
/Pages
/Kids
[
6
0
R
21
0
R
]
/Count
3
>>
endobj
xref
0 41
0000000002 65535 f
0000117809 00000 n
0000000003 00000 f
0000000000 00000 f
0000000016 00000 n
0000000160 00000 n
0000000207 00000 n
0000000373 00000 n
0000083202 00000 n
0000000730 00000 n
```

Ln : 1695 | UNIX | ANSI | OVR

my public prezo

public slide

public text

Edit out your page's reference

```
obj
15776
endobj
1
0
obj
<<
/Type
/Pages
/Kids
[
6
0
R



21
0
R
]
/Count
2
>>
endobj
xref
0 41
0000000002 65535 f
0000117809 00000 n
0000000003 00000 f
0000000000 00000 f
0000000016 00000 n
0000000160 00000 n
0000000207 00000 n
0000000373 00000 n
0000083202 00000 n
0000000730 00000 n
```

Ln : 1699 | UNIX | ANSI | OVR

and don't forget to update the pages' `/Count` ☺
(may lead to funny results)

Erasing a page with a tool

- tools such as PDFtk can operate on pages
  - ```
    D:\>pdftk "PDF Secrets.pdf" cat 1-3 5-end output no4.pdf
    ```

but:

- they don’t erase pages!
  - they extract the other pages

→ the whole page is lost

but the image contents (as objects) are still left!

and extractable!!
    ```

Erase overlapping element?

- remove paint/text operators from binary stream

Hint:

overlapping elements more likely at the end of the stream, as they were likely added last.

Operands	Operator	Description
—	**S**	Stroke the path.
—	**s**	Close and stroke the path. This operator shall have the same effect as the sequence h S.
—	**f**	Fill the path, using the nonzero winding number rule to determine the region to fill (see 8.5.3.3.2, "Nonzero Winding Number Rule"). Any subpaths that are open shall be implicitly closed before being filled.
—	**F**	Equivalent to **f**; included only for compatibility. Although PDF reader applications shall be able to accept this operator, PDF writer applications should use **f** instead.
—	**f***	Fill the path, using the even-odd rule to determine the region to fill (see 8.5.3.3.3, "Even-Odd Rule").
—	**B**	Fill and then stroke the path, using the nonzero winding number rule to determine the region to fill. This operator shall produce the same result as constructing two identical path objects, painting the first with **f** and the second with **S**. NOTE The filling and stroking portions of the operation consult different values of several graphics state parameters, such as the current colour. See also 11.7.4.4, "Special Path-Painting Considerations".
—	**B***	Fill and then stroke the path, using the even-odd rule to determine the region to fill. This operator shall produce the same result as **B**, except that the path is filled as if with **f*** instead of **f**. See also 11.7.4.4, "Special Path-Painting Considerations".
—	**b**	Close, fill, and then stroke the path, using the nonzero winding number rule to determine the region to fill. This operator shall have the same effect as the sequence h B. See also 11.7.4.4, "Special Path-Painting Considerations".
—	**b***	Close, fill, and then stroke the path, using the even-odd rule to determine the region to fill. This operator shall have the same effect as the sequence h B*. See also 11.7.4.4, "Special Path-Painting Considerations".
—	**n**	End the path object without filling or stroking it. This operator shall be a path-painting no-op, used primarily for the side effect of changing the current clipping path (see 8.5.4, "Clipping Path Operators").

paint operators

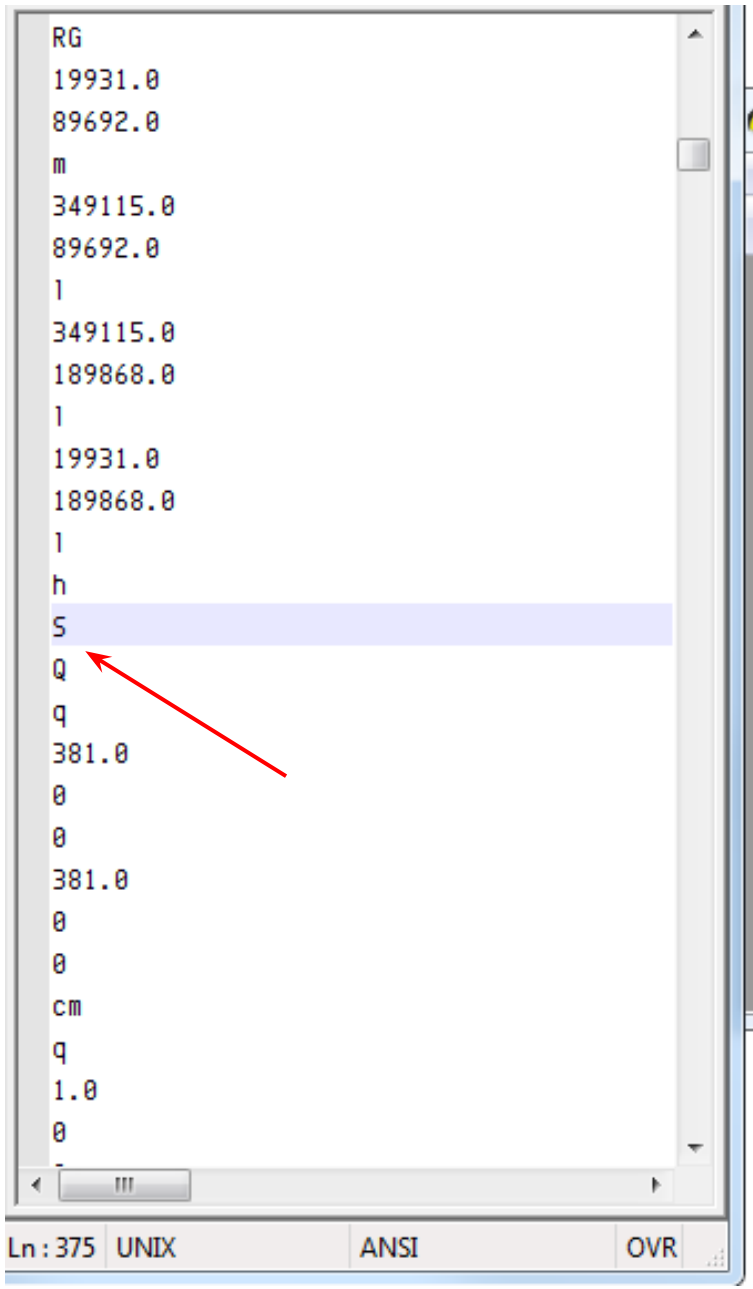
(PDF 32000-1:2008, page 135)

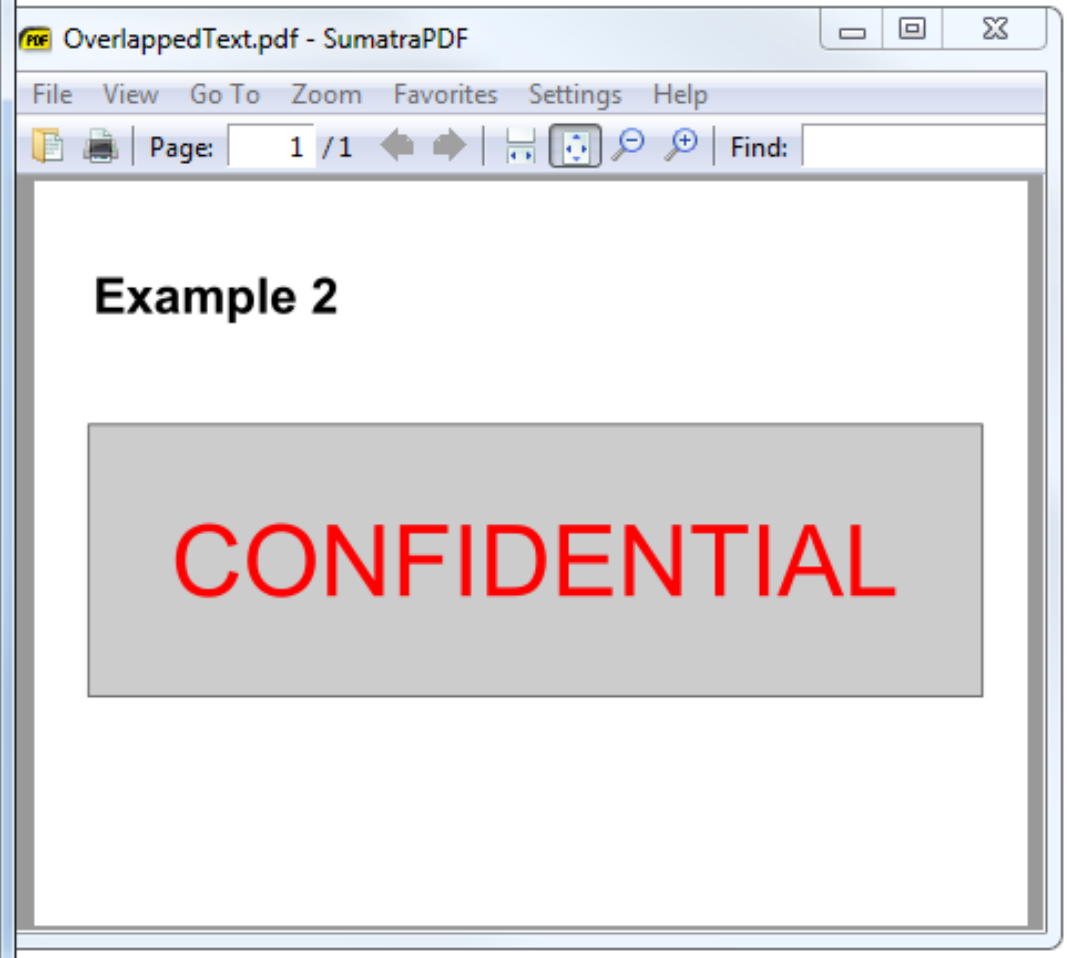
Operands	Operator	Description
string	**Tj**	Show a text string.
string	**'**	Move to the next line and show a text string. This operator shall have the same effect as the code T* *string* Tj
a_w a_c *string*	**"**	Move to the next line and show a text string, using a_w as the word spacing and a_c as the character spacing (setting the corresponding parameters in the text state). a_w and a_c shall be numbers expressed in unscaled text space units. This operator shall have the same effect as this code: a_w Tw a_c Tc *string* '
array	**TJ**	Show one or more text strings, allowing individual glyph positioning. Each element of *array* shall be either a string or a number. If the element is a string, this operator shall show the string. If it is a number, the operator shall adjust the text position by that amount; that is, it shall translate the text matrix, T_m. The number shall be expressed in thousandths of a unit of text space (see 9.4.4, "Text Space Details"). This amount shall be *subtracted* from the current horizontal or vertical coordinate, depending on the writing mode. In the default coordinate system, a positive adjustment has the effect of moving the next glyph painted either to the left or down by the given amount. Figure 46 shows an example of the effect of passing offsets to **TJ**.

text showing operators

(PDF 32000-1:2008, page 250-251)

Example: manually remove overlapping elements





take the uncompressed PDF

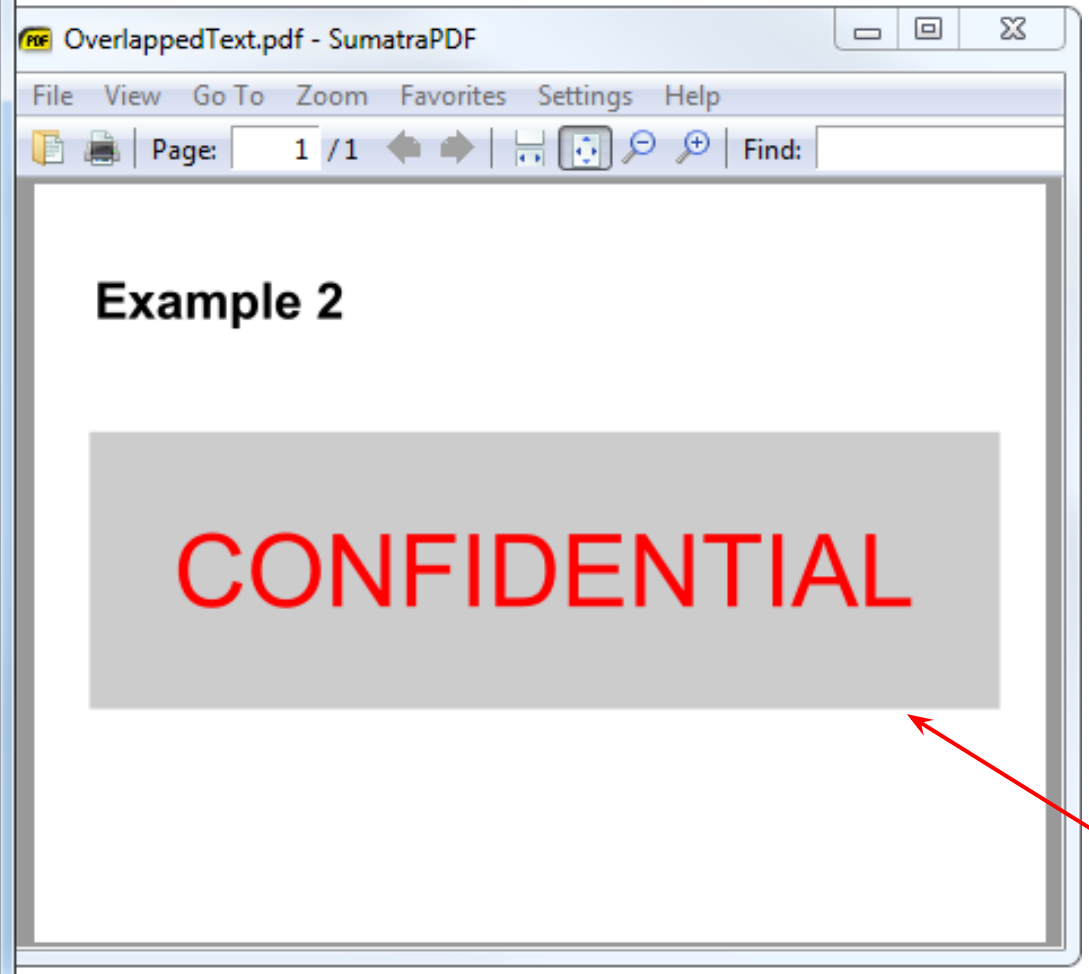
locate the `/Contents` stream object

locate the S (**S**troke path)

(you can search for **\nS\n**)

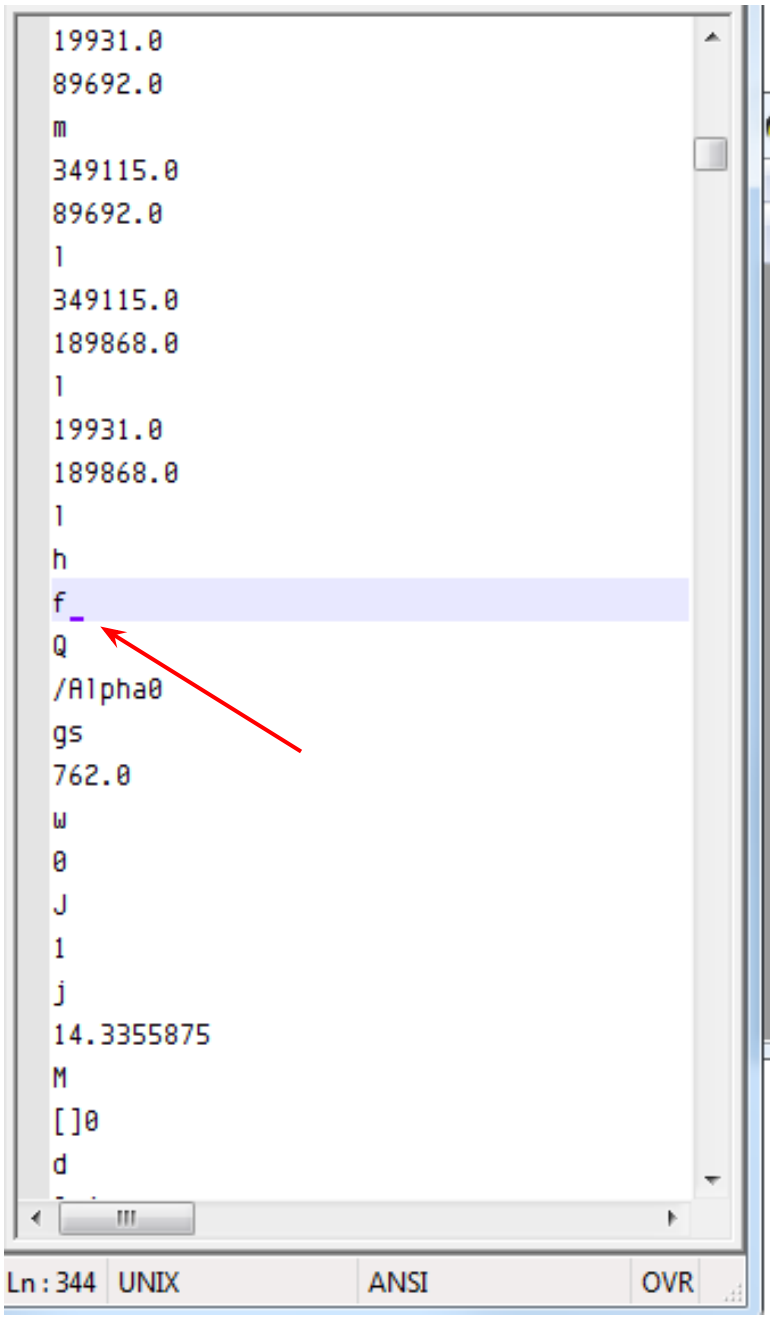
```
RG
19931.0
89692.0
m
349115.0
89692.0
l
349115.0
189868.0
l
19931.0
189868.0
l
h
Q
q
381.0
0
0
381.0
0
0
cm
q
1.0
0
```

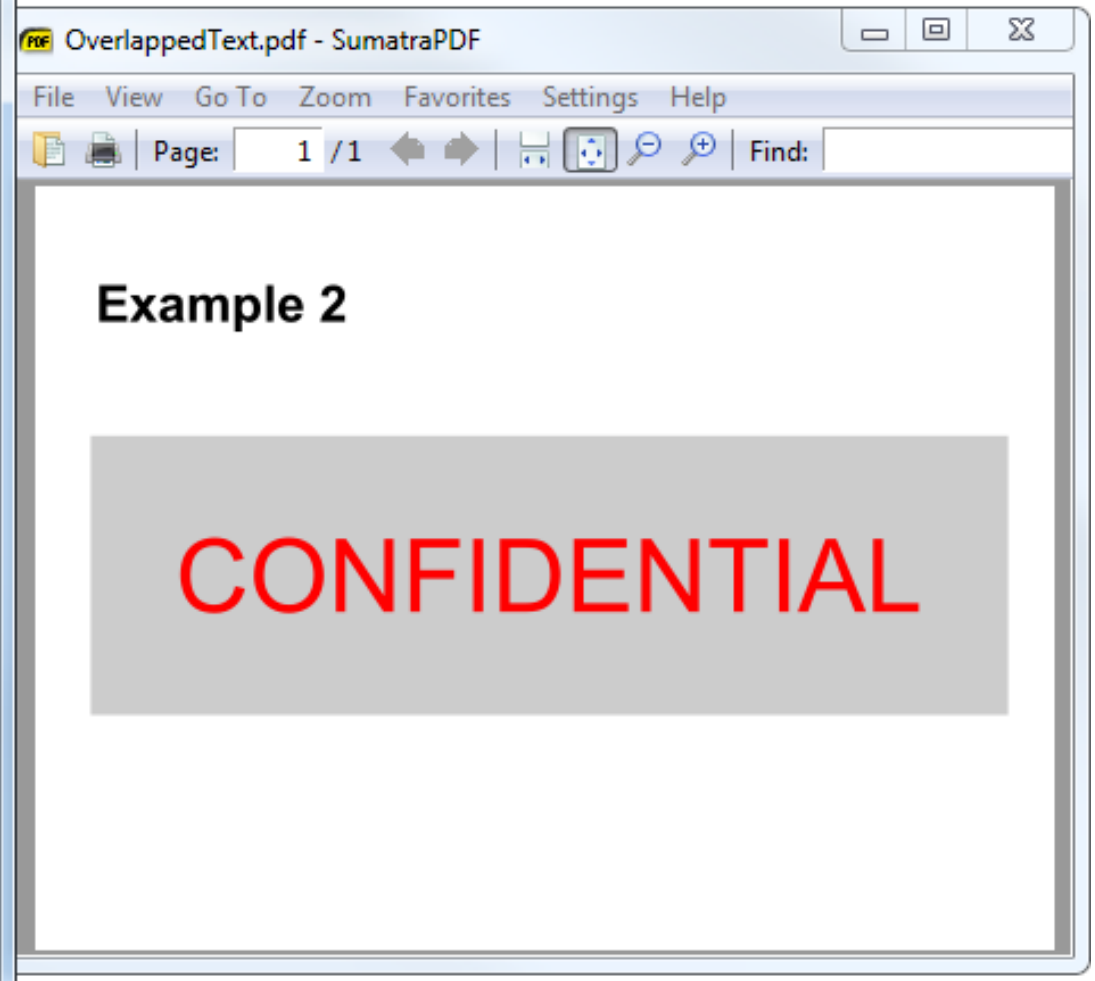
Ln : 375 | UNIX | ANSI | OVR



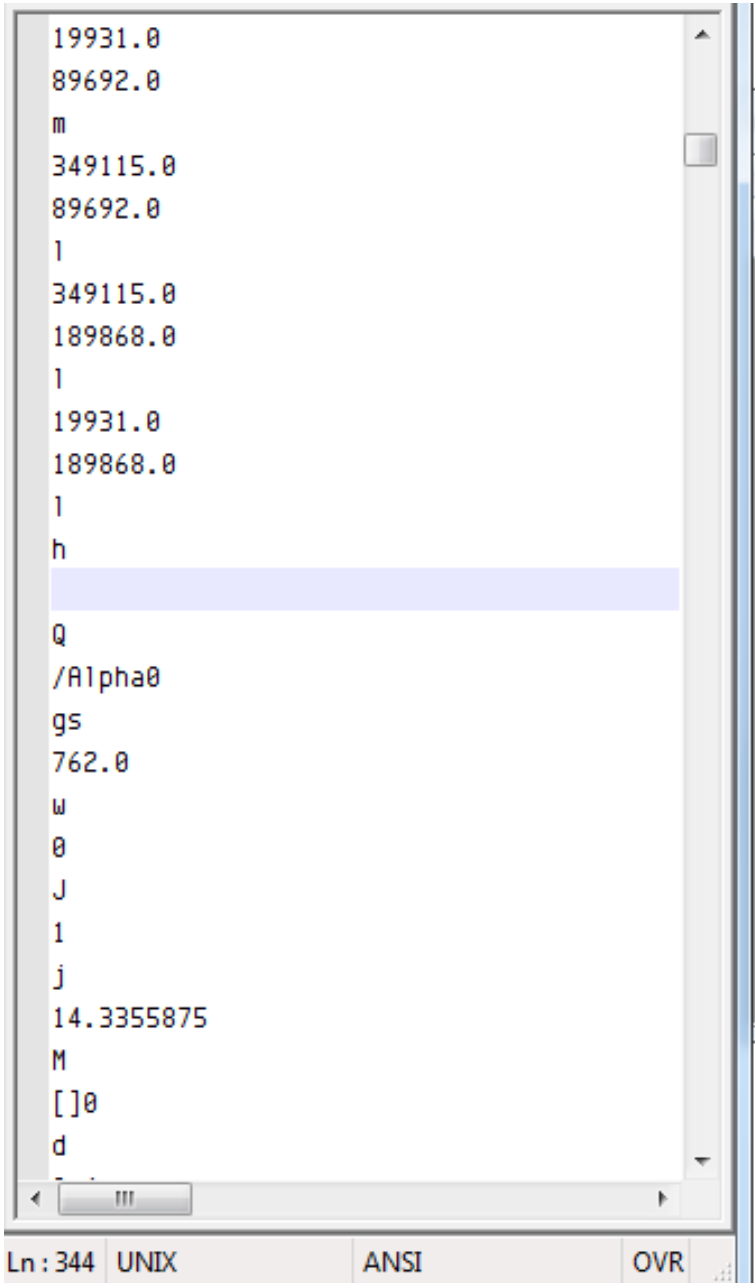
erase the S

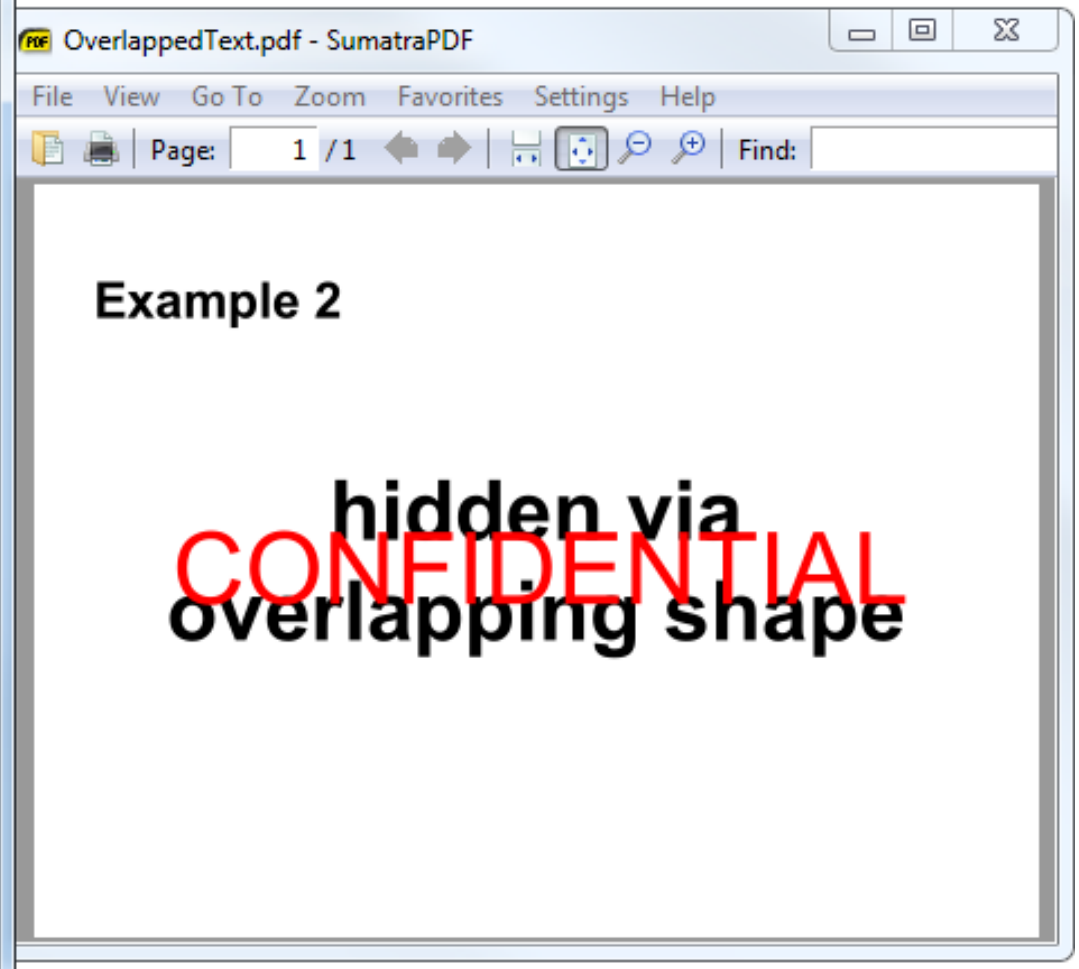
⇒ no more black border



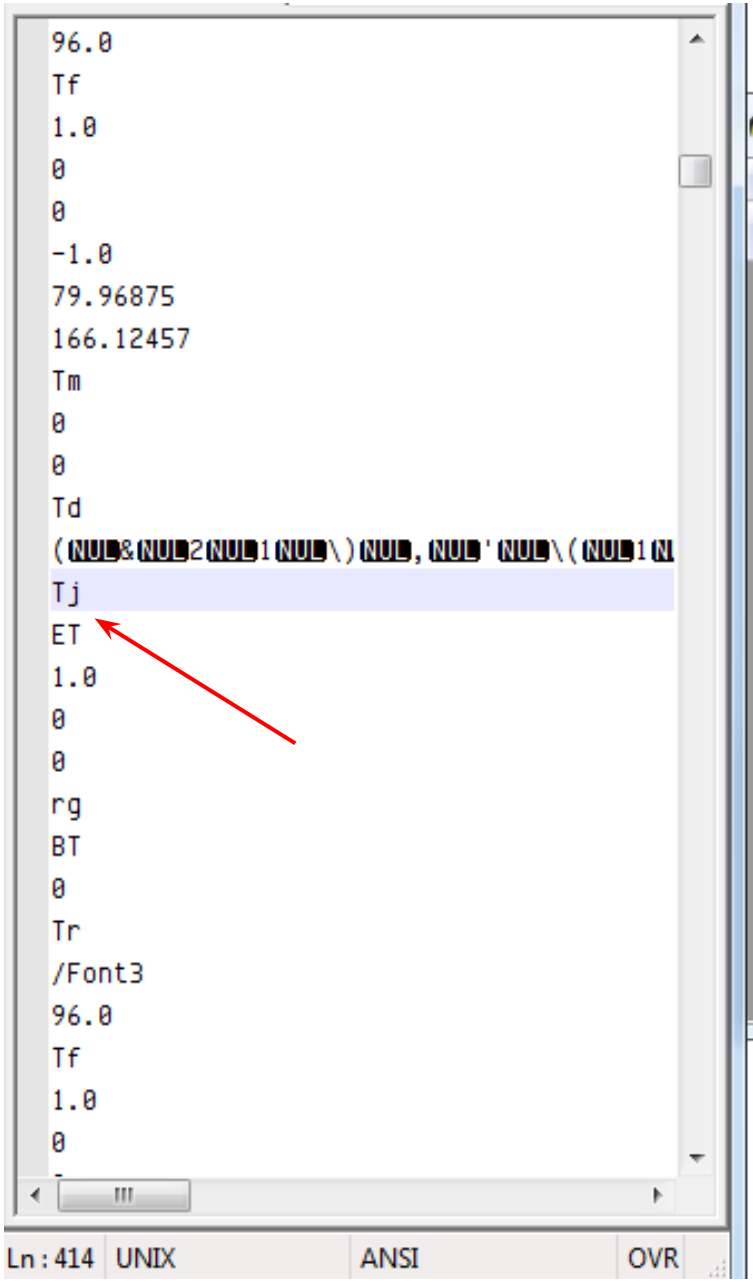


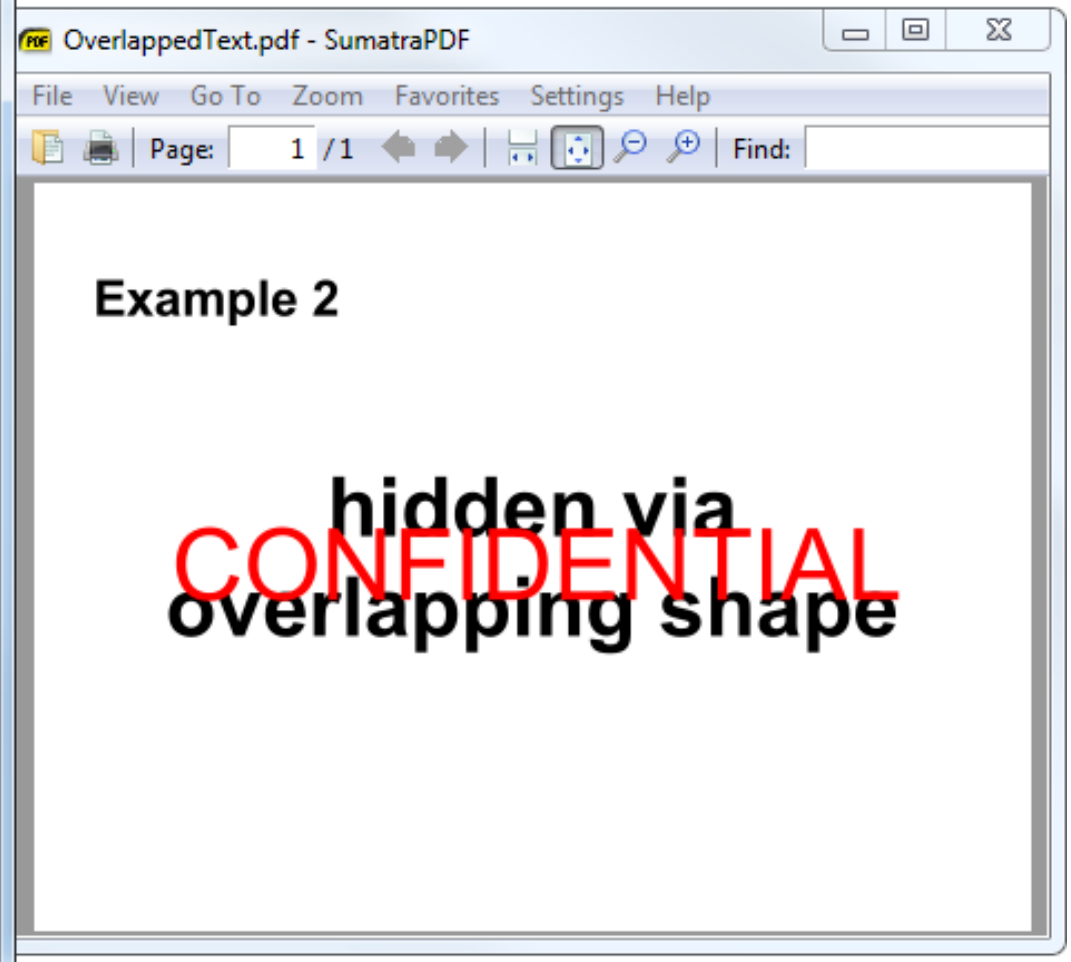
locate the f (path **F**illing)





⇒ no more gray surface

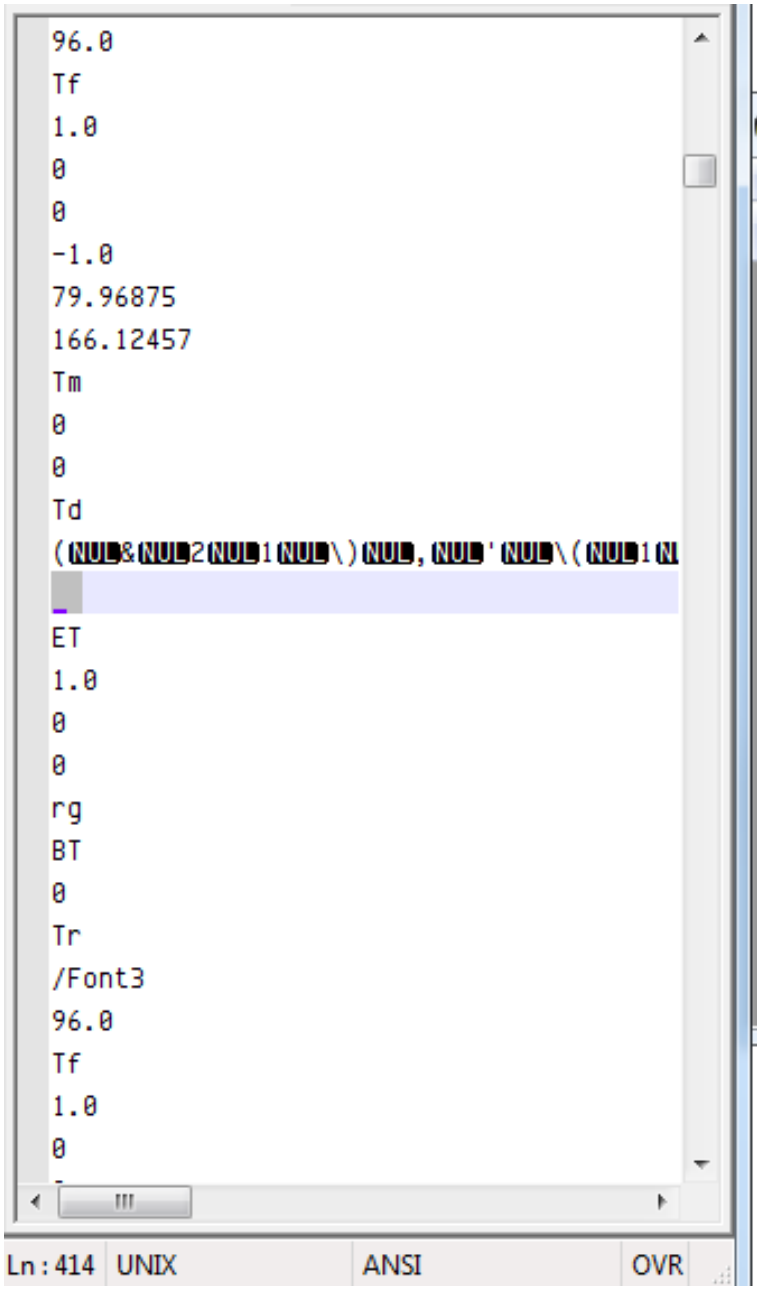


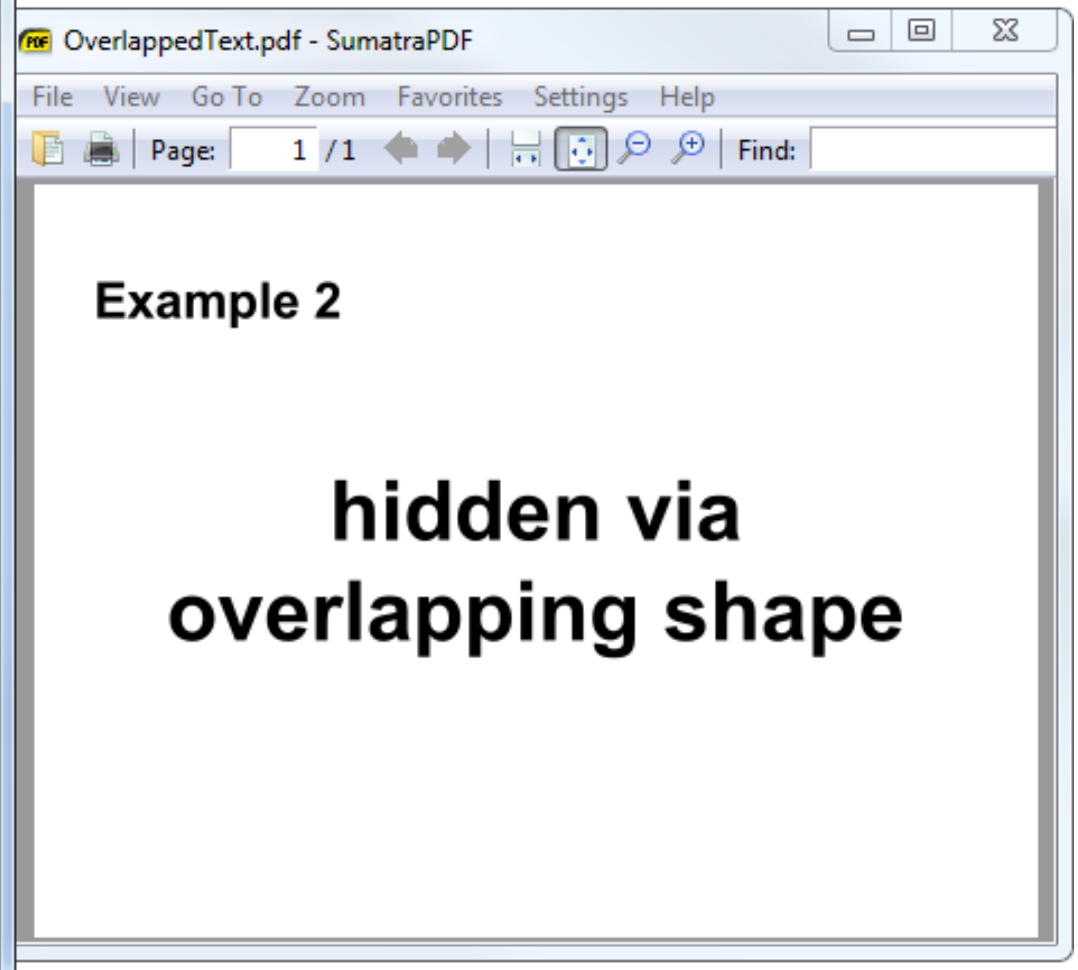


and the “obvious” `Tj` after the string `(...)`

Note: the letters are different, due to the font mapping

&→C, 2→O, 1→N...





→ no more hidden elements!

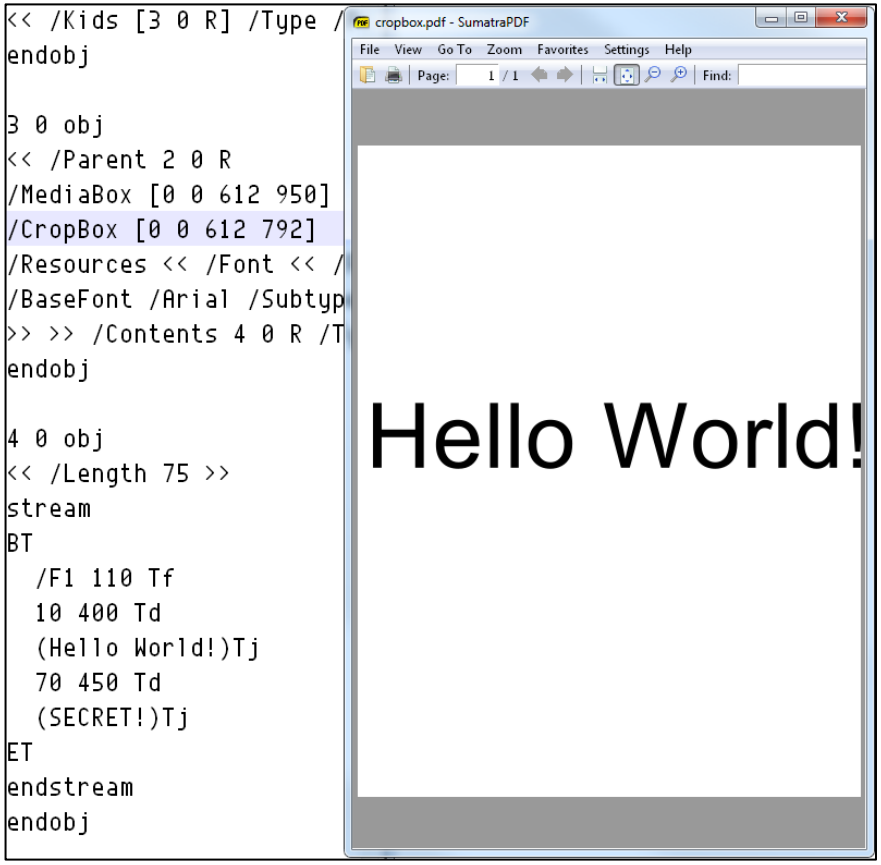
bonus: the operation can be easily automated!
(on all pages, etc…)

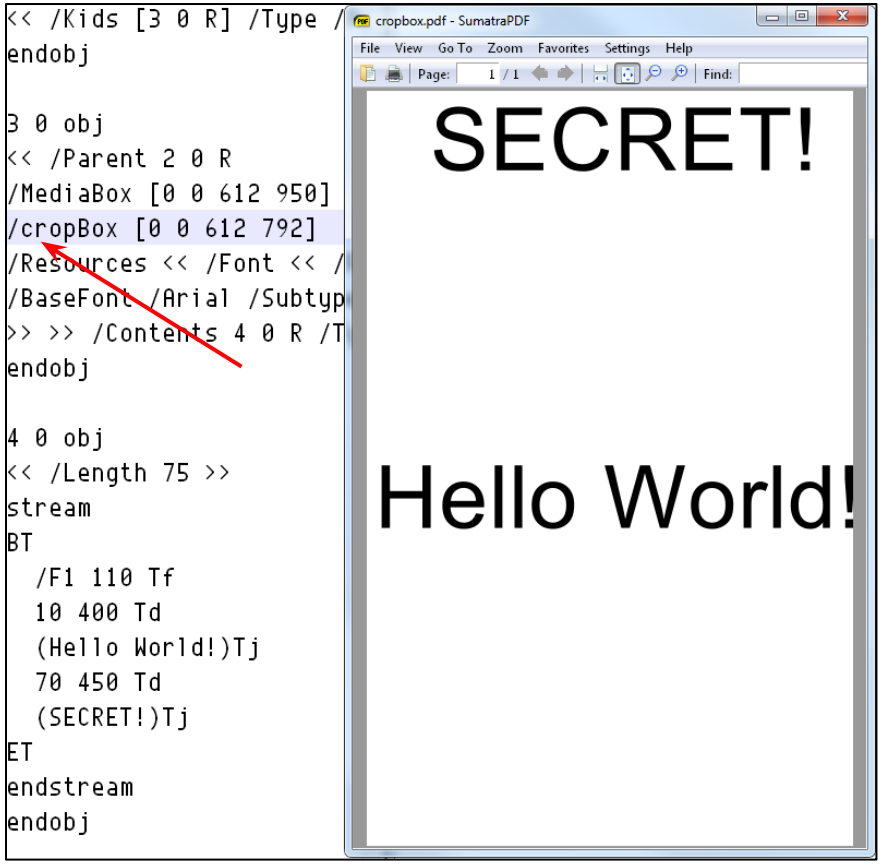
Page size tricks

- a page isn't just a /MediaBox :(
  - PDF is not so simple!
    - CropBox/BleedBox/TrimBox/ArtBox/...

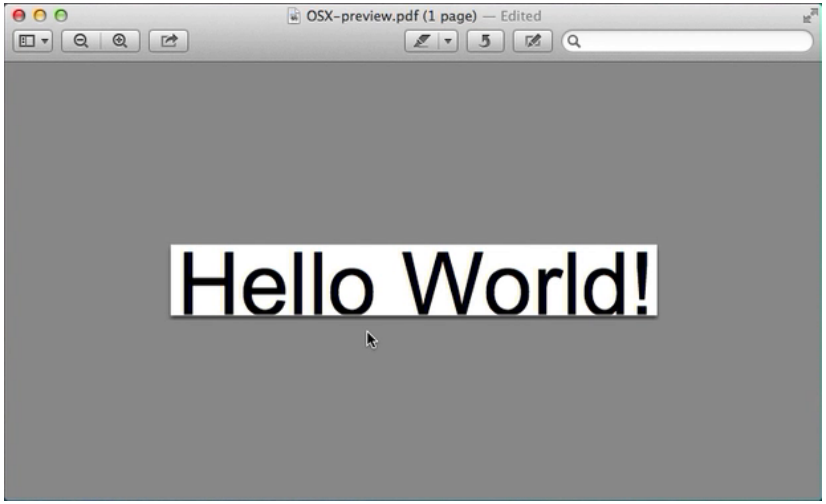
- What you see is `/CropBox`
  - Copy/Paste and (some) `pdftotext` respect that

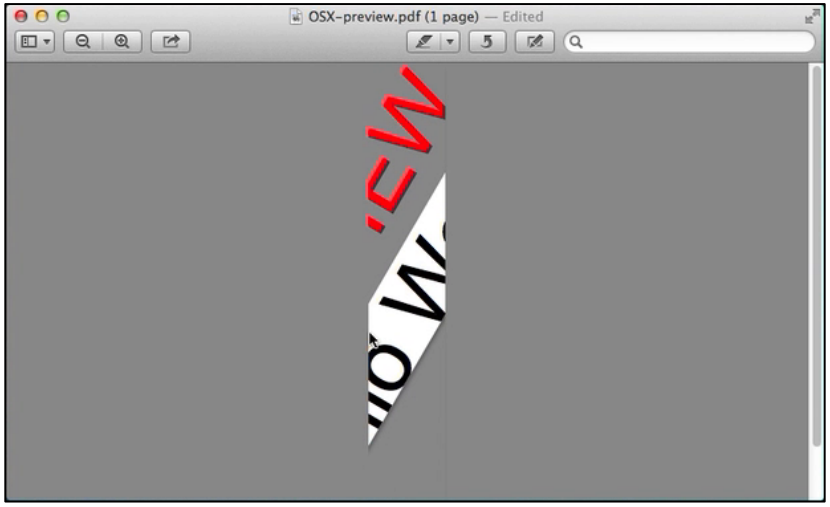
⇒ what is in `Mediabox` (but not `CropBox`)
is not extracted by tools or copy/paste





disable /CropBox to see the full contents

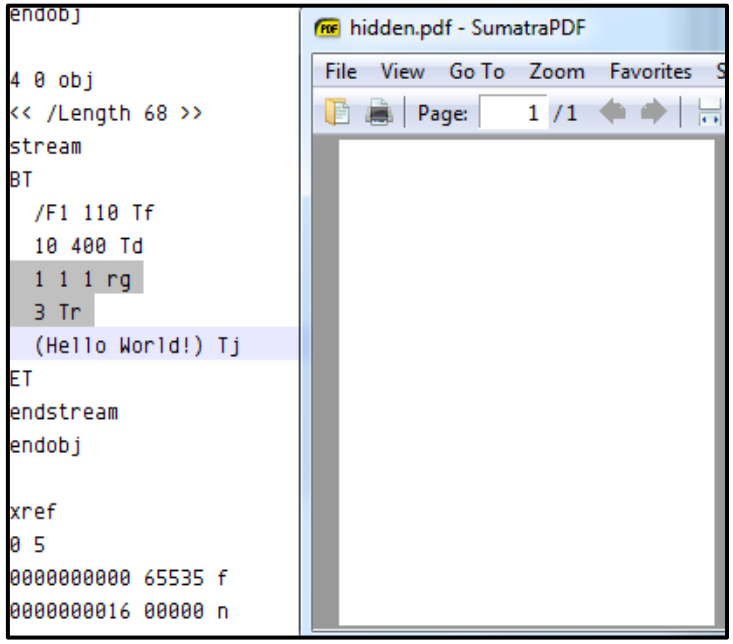


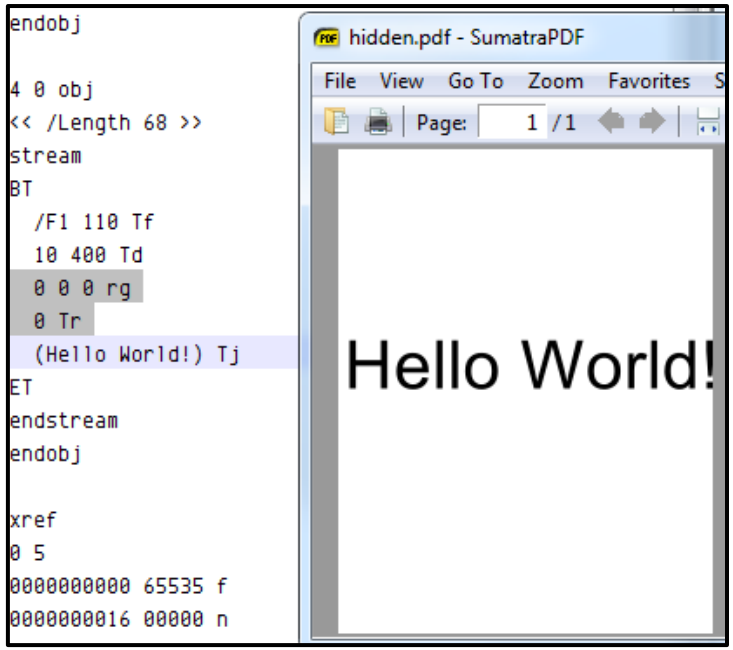


OS-X actually does a /CropBox when you copy/paste out of a PDF, and you can see the full original content by rotating the page.

Hidden text

- White color
  - `1 1 1 rg` (filling's color)
- text rendering mode
  - `3 Tr` = invisible
    - OCRs use it to store text





A more ‘deniable’ hiding

altering `/Kids` or the page’s `/Contents` work,

but there is another elegant solution:
incremental updates

PDF incremental updates

- not commonly used
  - required for signing
- but still supported by readers

the concept:

add another set of objects, xref, trailer, …

to update the objects' hierarchy

Example

a confidential object

with a secret stream object 4

to be hidden


```
%PDF-1.1
%âãÏÓ

1 0 obj
<< /Pages 2 0 R >>
endobj

2 0 obj
<< /Kids [3 0 R] /Type /Pages /Count 1 >>
endobj

3 0 obj
<< /Parent 2 0 R  /MediaBox [0 0 612 792]
/Resources << /Font << /F1 <<
/BaseFont /Arial /Subtype /Type1 /Type /Font>>
>> >> /Contents 4 0 R /Type /Page >>
endobj

4 0 obj
<< /Length 50 >>
stream
BT
  /F1 120 Tf
  10 400 Td
  (Top Secret) Tj
ET
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
0000000052 00000 n
0000000110 00000 n
0000000282 00000 n

trailer << /Size 5 /Root 1 0 R >>

startxref
385
%%EOF
```

New /Contents

append a new object 4

```
4 0 obj
<< /Length 52 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!) Tj
ET
endstream
endobj
```

Extra xref

append a new xref
that references it

```
xref
0 1
0000000000 65535 f
4 1
0000000551 00000 n
```

Extra trailer 1/2

- same `/Size` & `/Root`
- references the previous **xref** via `/Prev`
  (not the previous trailer)

```
trailer <<
    /Size 5
    /Root 1 0 R
    /Prev 385
>>
```

Extra trailer 2/2

points to the new **xref**

```
startxref
654
%%EOF
```

Result

⇒ different content !

restore content by cutting after the first %%EOF

Incremental update to hide page

use the same trick

to override `/Type /Pages`

```
…
%%EOF

1 0 obj
<<
/Type /Pages
/Kids [ 6 0 R 21 0 R]
/Count 2
>>
endobj

xref
0 1
0000000000 65535 f
1 1
0000118783 00000 n

trailer << /Size 41 /Root 4
0 R /Prev 117882 >>

startxref
118849
%%EOF
```

Actual leaks in the wild ?

in any PDF with `/Prev` in the `trailer`:
restore each intermediate version
by truncating after each `%%EOF`

DALLAS SEMICONDUCTOR MAXIM

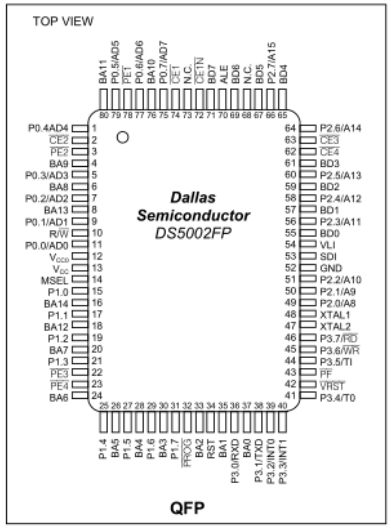
DS5002FP
Secure Microprocessor Chip

www.maxim-ic.com

GENERAL DESCRIPTION

The DS5002FP secure microprocessor chip is a secure version of the DS5001FP 128k soft microprocessor chip. In addition to the memory and I/O enhancements of the DS5001FP, the secure microprocessor chip incorporates the most sophisticated security features available in any processor. The security features of the DS5002FP include an array of mechanisms that are designed to resist all levels of threat, including observation, analysis, and physical attack. As a result, a massive effort is required to obtain any information about memory contents. Furthermore, the "soft" nature of the DS5002FP allows frequent modification of the secure information, thereby minimizing the value of any secure information obtained by such a massive effort.

PIN CONFIGURATION



FEATURES

- **8051-Compatible Microprocessor for Secure/Sensitive Applications**
  - Access 32kB, 64kB, or 128kB of NV SRAM for Program and/or Data Storage
  - In-System Programming Through On-Chip Serial Port
  - Can Modify Its Own Program or Data Memory in the End System
- **Firmware Security Features**
  - Memory Stored in Encrypted Form
  - Encryption Using On-Chip 64-Bit Key
  - Automatic True Random Key Generator
  - Self Destruct Input (SDI)
  - Optional Top Coating Prevents Microprobe (DS5002FPM)
  - Improved Security Over Previous Generations
  - Protects Memory Contents from Piracy
- **Crash-Proof Operation**
  - Maintains All Nonvolatile Resources for Over 10 Years in the Absence of Power
  - Power-Fail Reset
  - Early Warning Power-Fail Interrupt
  - Watchdog Timer

ORDERING INFORMATION

PART	TEMP RANGE	INTERNAL MICRO PROBE SHIELD	PIN-PACKAGE
DS5002FPM-16	0°C to +70°C	Yes	80 QFP
DS5002FPM-16+	0°C to +70°C	Yes	80 QFP
DS5002FMN-16	-40°C to +85°C	Yes	80 QFP
DS5002FMN-16+	-40°C to +85°C	Yes	80 QFP

+ Denotes a Pb-free/RoHS-compliant device.

Selector Guide appears at end of data sheet.

Note: *Some revisions of this device may incorporate deviations from published specifications known as errata. Multiple revisions of any device may be simultaneously available through various sales channels. For information about device errata, click here: www.maxim-ic.com/errata.*

REV: 072806

DALLAS SEMICONDUCTOR MAXIM

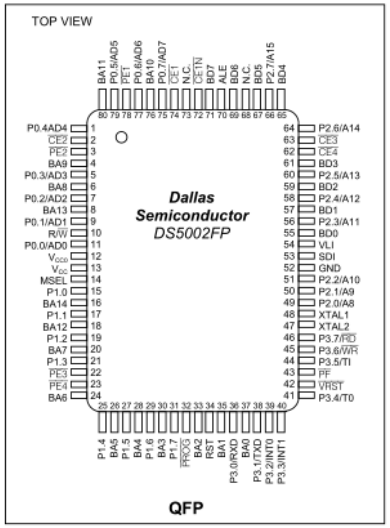
DS5002FP
Secure Microprocessor Chip

www.maxim-ic.com

GENERAL DESCRIPTION

The DS5002FP secure microprocessor chip is a secure version of the DS5001FP 128k soft microprocessor chip. In addition to the memory and I/O enhancements of the DS5001FP, the secure microprocessor chip incorporates the most sophisticated security features available in any processor. The security features of the DS5002FP include an array of mechanisms that are designed to resist all levels of threat, including observation, analysis, and physical attack. As a result, a massive effort is required to obtain any information about memory contents. Furthermore, the "soft" nature of the DS5002FP allows frequent modification of the secure information, thereby minimizing the value of any secure information obtained by such a massive effort.

PIN CONFIGURATION



FEATURES

- **8051-Compatible Microprocessor for Secure/Sensitive Applications**
  - Access 32kB, 64kB, or 128kB of NV SRAM for Program and/or Data Storage
  - In-System Programming Through On-Chip Serial Port
  - Can Modify Its Own Program or Data Memory in the End System
- **Firmware Security Features**
  - Memory Stored in Encrypted Form
  - Encryption Using On-Chip 64-Bit Key
  - Automatic True Random Key Generator
  - Self Destruct Input (SDI)
  - Optional Top Coating Prevents Microprobe (DS5002FPM)
  - Improved Security Over Previous Generations
  - Protects Memory Contents from Piracy
- **Crash-Proof Operation**
  - Maintains All Nonvolatile Resources for Over 10 Years in the Absence of Power
  - Power-Fail Reset
  - Early Warning Power-Fail Interrupt
  - Watchdog Timer

ORDERING INFORMATION

PART	TEMP RANGE	INTERNAL MICRO PROBE SHIELD	PIN-PACKAGE
DS5002FP-16	0°C to +70°C	No	80 QFP
DS5002FP+16	0°C to +70°C	No	80 QFP
DS5002FPM-16	0°C to +70°C	Yes	80 QFP
DS5002FPM+16	0°C to +70°C	Yes	80 QFP
DS5002FP-16N	-40°C to +85°C	No	80 QFP
DS5002FP+16N	-40°C to +85°C	No	80 QFP
DS5002FMN-16	-40°C to +85°C	Yes	80 QFP
DS5002FMN+16	-40°C to +85°C	Yes	80 QFP

+ Denotes a Pb-free/RoHS-compliant device.

Selector Guide appears at end of data sheet.

Note: *Some revisions of this device may incorporate deviations from published specifications known as errata. Multiple revisions of any device may be simultaneously available through various sales channels. For information about device errata, click here: www.maxim-ic.com/errata.*

REV: 090805

incremental PDF found in the wild
(removed parts, incorrect page number)

REVISION HISTORY

REVISION	DESCRIPTION
112795	Original release.
073096	Change V_{CC02} specification from V_{LI} - 0.5 to V_{LI} - 0.65 (PCN F62501). Update mechanical specifications.
111996	Change V_{CC01} from V_{CC} - 0.3 to V_{CC} - 0.35.
061297	$\overline{PF}$ signal moved from V_{OL2} test specification to V_{OL1}. PCN No. (D72502). AC characteristics for battery-backed SDI pulse specification added.
051499	Reduced absolute maximum voltage to V_{CC} + 0.5V. Added note clarifying storage temperature specification is for nonbattery-backed state. Deleted I_{BAT} specification (Duplicate of I_{LI} specification). Changed RRE min (industrial temp range) from 40kΩ to 30kΩ. Changed V_{PFW} max (industrial temp range) from 4.5V to 4.6V. Added industrial specification for I_{LI}. Reduced t_{CE1HOV} and t_{CEHDV} from 10ns to 0ns.
052599	Minor revisions and approval.
062102	Update V_{CCO} and I_{CCO1} specifications to reflect 0.45V internal voltage drop instead of 0.35V.
100102	Ordering information updated.
030403	Reset Trip Point in Stop Mode (DC Characteristics) with BAT = 3.0V was changed to 3.3V (original issue was 3.3V).
070605	Added Pb-free part numbers to Ordering Information and Selector Guide. Added Operating Voltage specification. (This is not a new specification because operating voltage is implied in the testing limits, but rather a clarification.) Updated Absolute Maximum soldering temperature to reference JEDEC standard.
090805	In the *AC Characteristics—SDI Pin* table, changed t_{SPR} MAX (in active mode) from 2µs to 1.3µs. This change is only to correct a documentation error, and does not reflect a change in device operation or any change in testing.
072806	Removed products from Ordering Information table that do not contain internal micro probe shields.

Maxim Integrated Products, 120 San Gabriel Drive, Sunnyvale, CA 94086 408-737-7600

DS5002FP Secure Microprocessor Chip

REVISION HISTORY

REVISION	DESCRIPTION
112795	Original release.
073096	Change V_{CC02} specification from V_{LI} - 0.5 to V_{LI} - 0.65 (PCN F62501). Update mechanical specifications.
111996	Change V_{CC01} from V_{CC} - 0.3 to V_{CC} - 0.35.
061297	$\overline{PF}$ signal moved from V_{OL2} test specification to V_{OL1}. PCN No. (D72502). AC characteristics for battery-backed SDI pulse specification added.
051499	Reduced absolute maximum voltage to V_{CC} + 0.5V. Added note clarifying storage temperature specification is for nonbattery-backed state. Deleted I_{BAT} specification (Duplicate of I_{LI} specification). Changed RRE min (industrial temp range) from 40kΩ to 30kΩ. Changed V_{PFW} max (industrial temp range) from 4.5V to 4.6V. Added industrial specification for I_{LI}. Reduced t_{CE1HOV} and t_{CEHDV} from 10ns to 0ns.
052599	Minor revisions and approval.
062102	Update V_{CCO} and I_{CCO1} specifications to reflect 0.45V internal voltage drop instead of 0.35V.
100102	Ordering information updated.
030403	Reset Trip Point in Stop Mode (DC Characteristics) with BAT = 3.0V was changed to 3.3V (original issue was 3.3V).
070605	Added Pb-free part numbers to Ordering Information and Selector Guide. Added Operating Voltage specification. (This is not a new specification because operating voltage is implied in the testing limits, but rather a clarification.) Updated Absolute Maximum soldering temperature to reference JEDEC standard.
090805	In the *AC Characteristics—SDI Pin* table, changed t_{SPR} MAX (in active mode) from 2µs to 1.3µs. This change is only to correct a documentation error, and does not reflect a change in device operation or any change in testing.

25 of 25

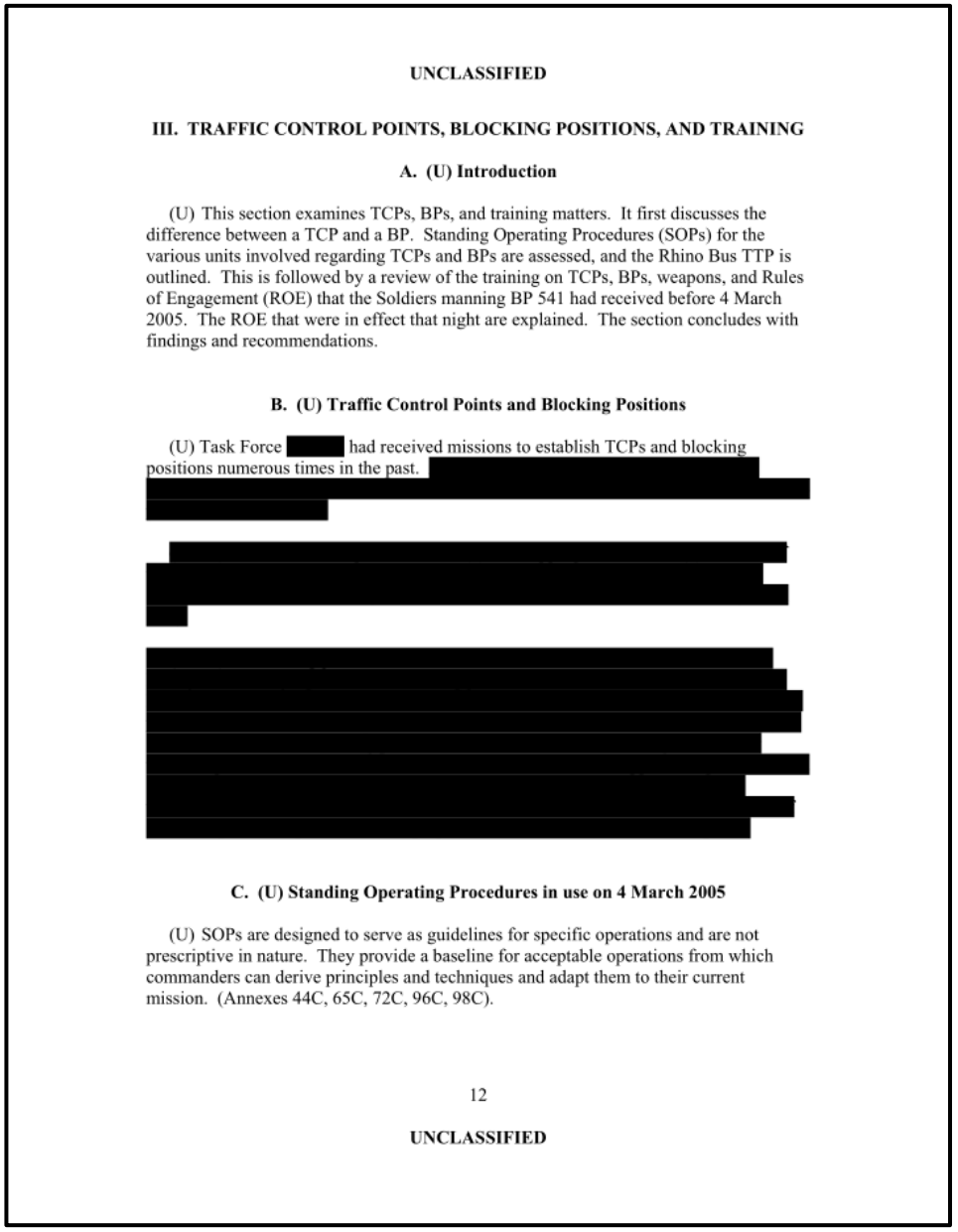
Maxim/Dallas Semiconductor cannot assume responsibility for use of any circuitry other than circuitry entirely embodied in a Maxim/Dallas Semiconductor product. No circuit patent licenses are implied. Maxim/Dallas Semiconductor reserves the right to change the circuitry and specifications without notice at any time.

Maxim Integrated Products, 120 San Gabriel Drive, Sunnyvale, CA 94086 408-737-7600

"Printed USA"

real examples

1. decompress
2. locate page
3. locate content
4. locate operator
5. disable all operators



UNCLASSIFIED

III. TRAFFIC CONTROL POINTS, BLOCKING POSITIONS, AND TRAINING

A. (U) Introduction

(U) This section examines TCPs, BPs, and training matters. It first discusses the difference between a TCP and a BP. Standing Operating Procedures (SOPs) for the various units involved regarding TCPs and BPs are assessed, and the Rhino Bus TTP is outlined. This is followed by a review of the training on TCPs, BPs, weapons, and Rules of Engagement (ROE) that the Soldiers manning BP 541 had received before 4 March 2005. The ROE that were in effect that night are explained. The section concludes with findings and recommendations.

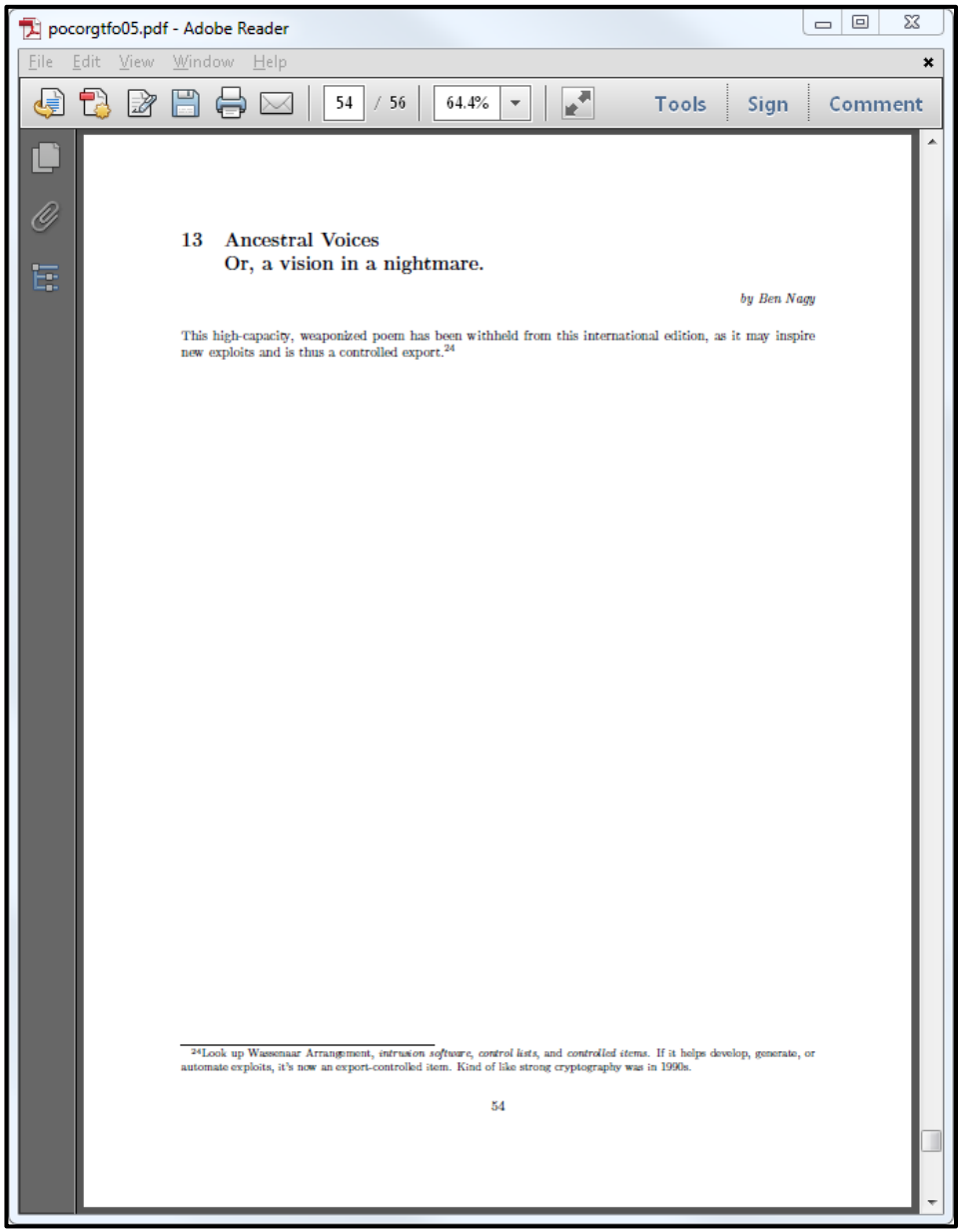
B. (U) Traffic Control Points and Blocking Positions

(U) Task Force [redacted] had received missions to establish TCPs and blocking positions numerous times in the past. [redacted]

C. (U) Standing Operating Procedures in use on 4 March 2005

(U) SOPs are designed to serve as guidelines for specific operations and are not prescriptive in nature. They provide a baseline for acceptable operations from which commanders can derive principles and techniques and adapt them to their current mission. (Annexes 44C, 65C, 72C, 96C, 98C).

1. restore structure
2. decompress
3. locate *
4. modify operator



Conclusion

Conclusion

- the PDF file format is awkward
  - not too complex if you just want to hide/reveal secrets
- be careful when removing sensitive elements!
  - quite easy to check if elements are still removed or not
  - overlapping DOESN’T work
- hiding and recovering elements is ‘easy’
  - content is still there!

Suggestions?

I’m interested in:

- hiding technics
- automated revealing technics
- documents that are a pain to ‘rebuild’
  - split fonts in small paths ?
  - licensed fonts are converted to glyphs
    ⇒ no more text

ACK

@pdfkungfoo

@Daeinar @veorq @_Quack1 @MunrekFR @dominicgs @mwgamera @kevinallix @munin @kristamonster @ClaudioAlbertin @push_pnx @JHeguia @doegox @gynvael @nst021 @iamreddave @chrisnklein

@angealbertini
corkami.com

PDF secrets