PDF secrets

hiding & revealing secrets in PDF documents

RaumZeitLabor

Mannheim Germany

Ange Albertini

2014/05/17

reverse engineering

&

VISUAL DOCUMENTATIONS

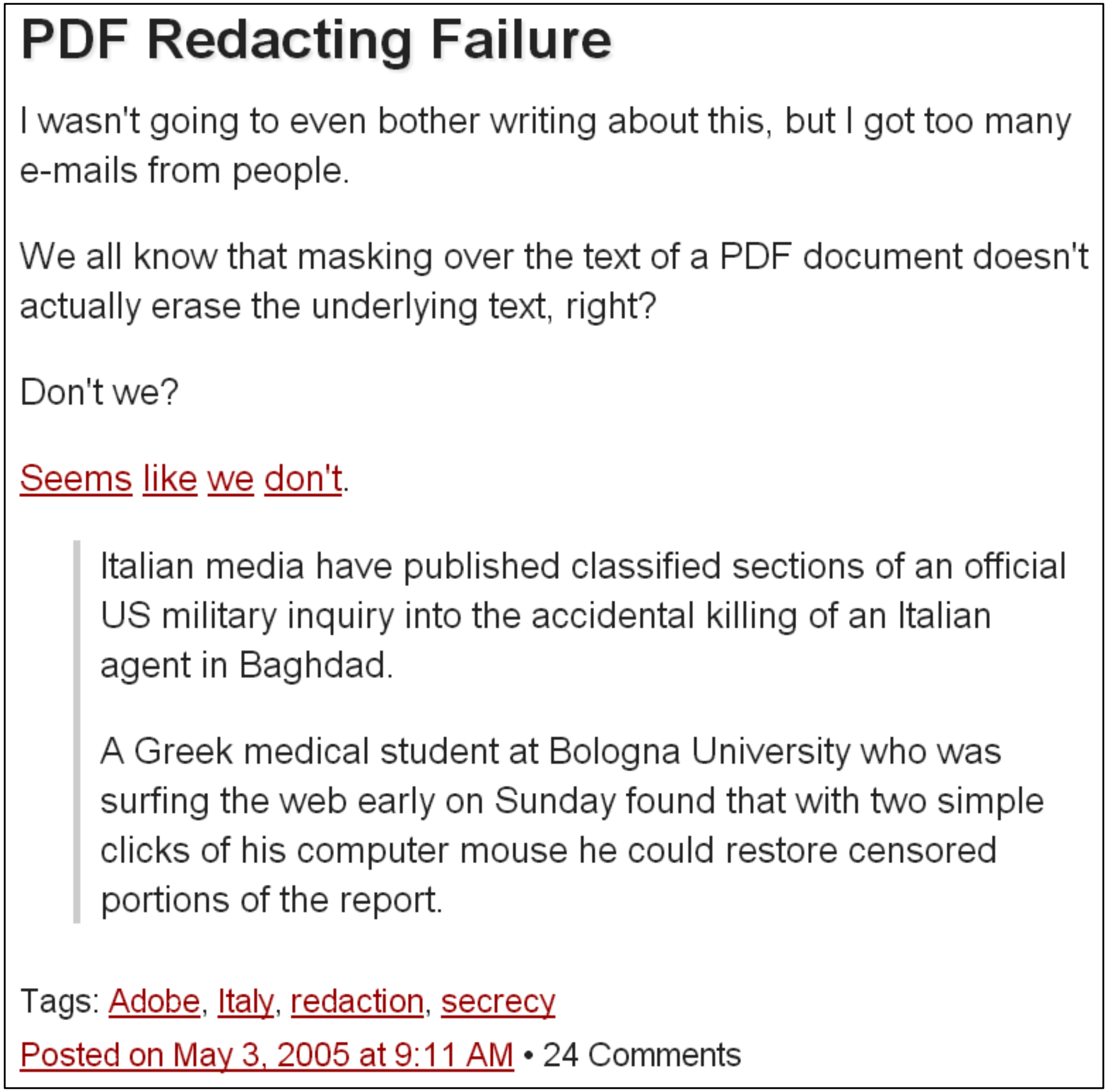
corkami.com

The problem

You need to remove sensitive elements of a PDF document for public release

are they actually removed ?

can someone reveal your secrets ?



PDF Redacting Failure

I wasn't going to even bother writing about this, but I got too many e-mails from people.

We all know that masking over the text of a PDF document doesn't actually erase the underlying text, right?

Don't we?

Seems like we don't.

> Italian media have published classified sections of an official US military inquiry into the accidental killing of an Italian agent in Baghdad.
>
> A Greek medical student at Bologna University who was surfing the web early on Sunday found that with two simple clicks of his computer mouse he could restore censored portions of the report.

Tags: Adobe, Italy, redaction, secrecy

Posted on May 3, 2005 at 9:11 AM • 24 Comments

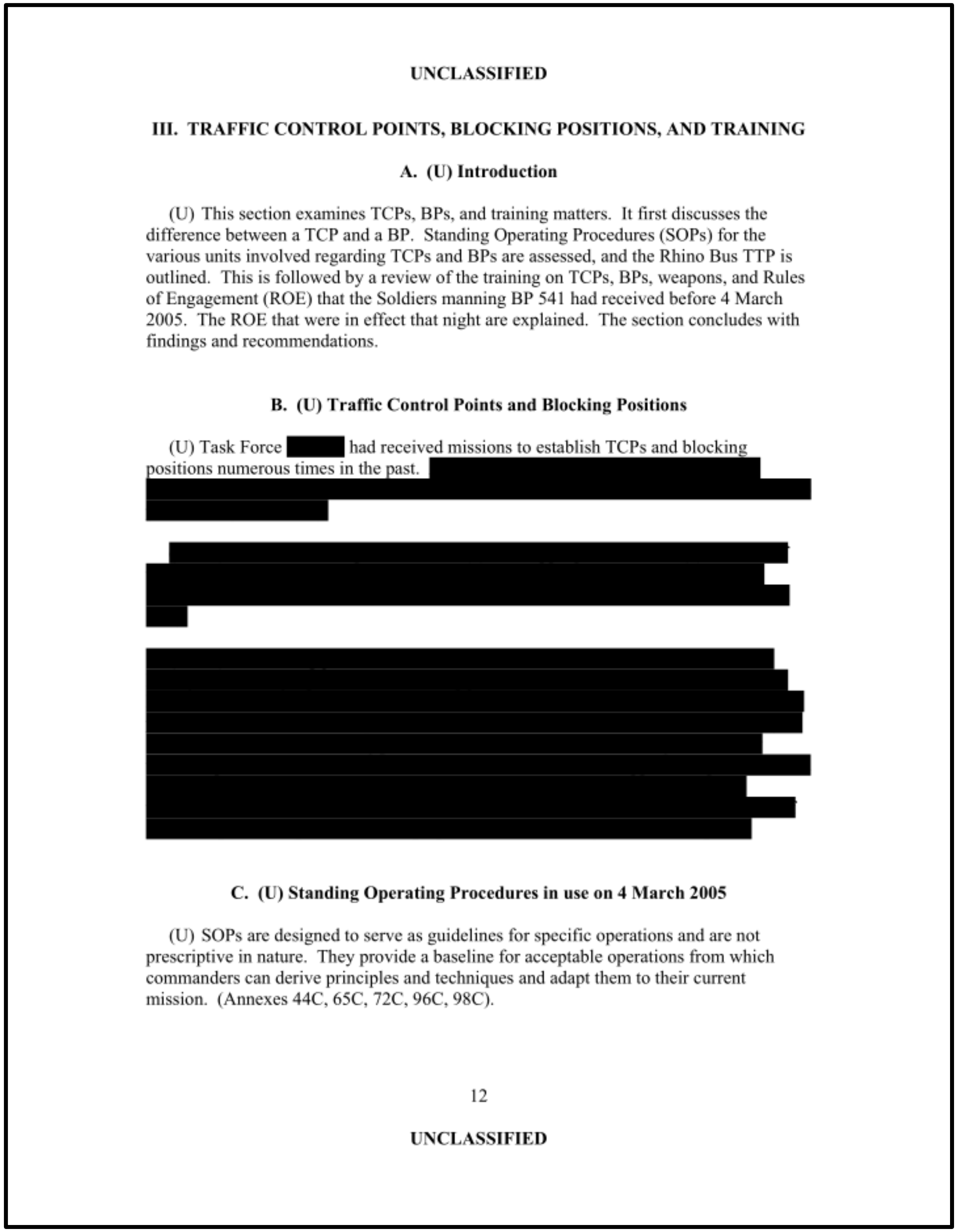
https://www.schneier.com/blog/archives/2005/05/pdf_radacting_f.html

It’s not a new fact

There are plenty of real examples

You just need to:

1. uncompress the PDF
2. remove all “ re\n” occurences
   (“re” = **re**ctangle operator)



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]

[redacted]

[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).

12

UNCLASSIFIED

http://download.repubblica.it/pdf/rapportousacalipari.pdf
seen in its metadata: “EmailSubject (Another Redact Job For You)”

the topic wasn’t really covered *technically*

AFAIK

The reverse problem

You need to carry a sensible PDF,
or exfiltrate some information:

Can you convincingly pretend
that it was a mistake,
and yet easily re-enable the contents?

...and, more importantly...

it still makes it an interesting exercise
to learn and experiment with PDF internals ☺

...and it might also be useful for a CTF steganography challenge...

it’s about hiding parts of the PDF document

not hiding data in a PDF file
+ nothing reader-specific

General outline of this talk

3 relatively independent parts:

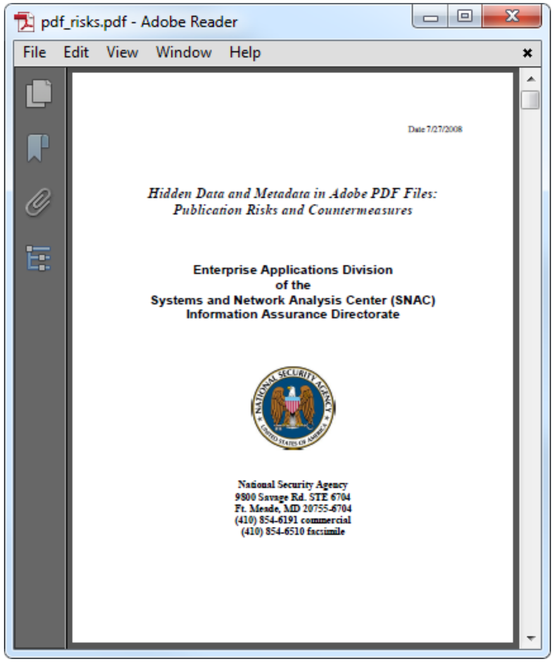
1. a non-technical approach
2. a basic introduction to the PDF file format
3. a technical perspective

a non-technical approach

Part I / III

What about that NSA doc ?

there is an NSA document on the topic.
worth a read, but Adobe Acrobat (Pro) only



Date 7/27/2008

Hidden Data and Metadata in Adobe PDF Files:
Publication Risks and Countermeasures

Enterprise Applications Division
of the
Systems and Network Analysis Center (SNAC)
Information Assurance Directorate

National Security Agency
9800 Savage Rd. STE 6704
Ft. Meade, MD 20755-6704
(410) 854-6191 commercial
(410) 854-6510 facsimile

http://www.nsa.gov/ia/_files/app/pdf_risks.pdf

Preamble

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

http://pdf.corkami.com

Outline

1. the problem (introduction)
2. outline
   a. see Google “recursion” ☺
3. examples
   a. color
      i. forgotten text
   b. overlapped text
   c. secured documents
      i. bypassing security
   d. overlapped image
      i. extracting image
4. Conclusion

So, you tried to hide elements in a PDF...

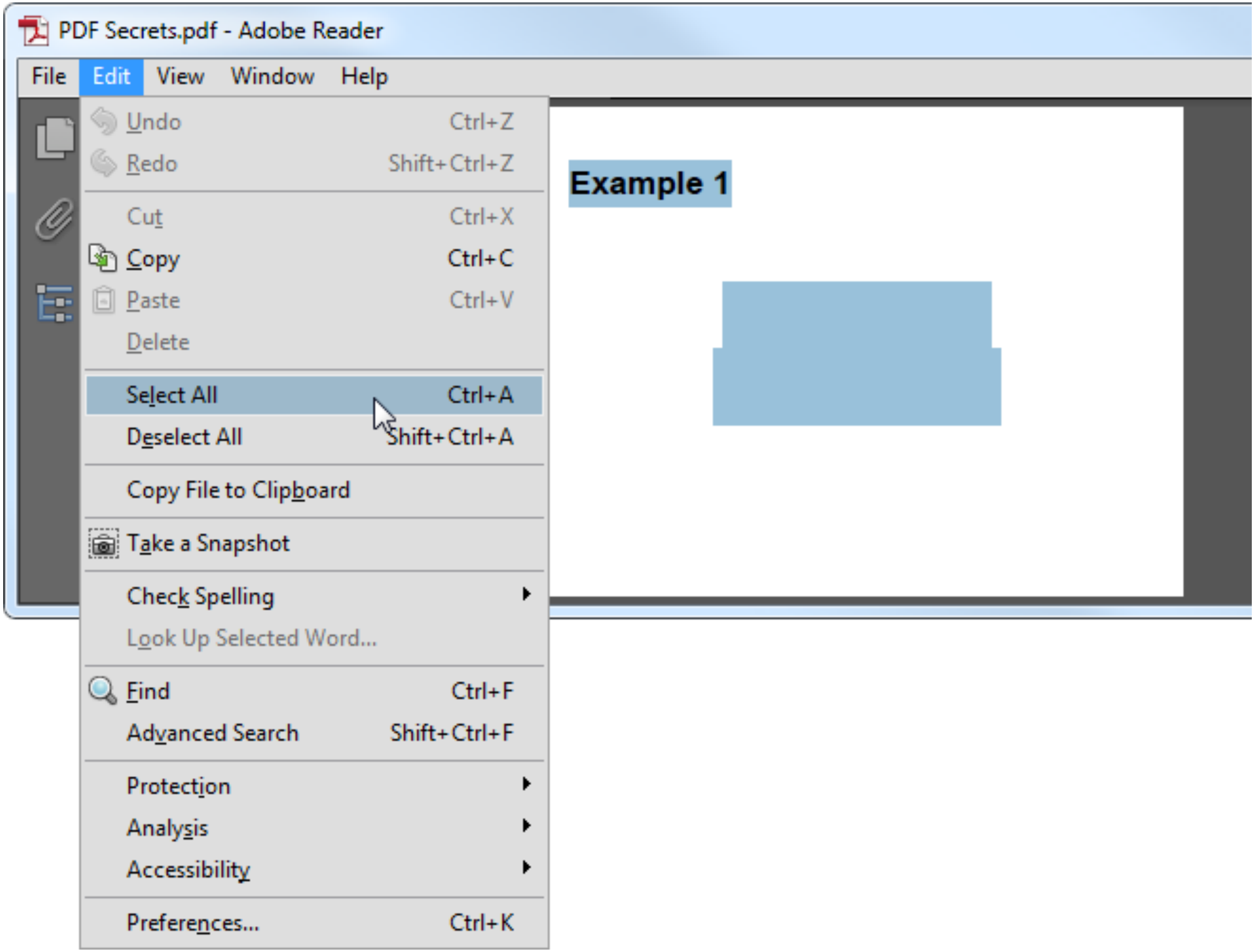
“well, I don’t see them anymore”

try with the next slide:

nothing is visible… and yet...

1. “Select All” text with your favorite PDF viewer
2. Copy and paste in a text editor

Example: color



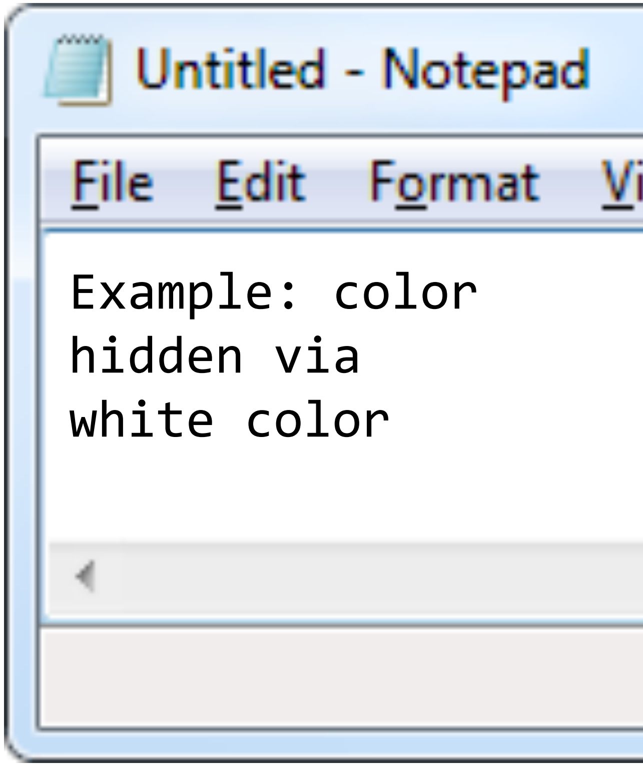
Example 1

hint

It worked, right?

you can’t see the text,
but it’s still on the page

→ the software can select it



Example: color
hidden via
white color

Btw...

this can lead to unexpected results,
so be careful before publishing slides,
even if you think you have nothing to remove

try with next slide ☺

Example: forgotten text

HyperVortex 1.0
a publication software

Roberto Martinez

Oops

maybe it wasn’t a secret to be removed,
but’s still there!

put extra hidden content for easier indexing

god, I hate making slides!!!
Example: forgotten text
HyperVortex 1.0
a publication software
title
Roberto Martinez
authors
insert stupid footer here -- LaTeX sucks!!!

Another try

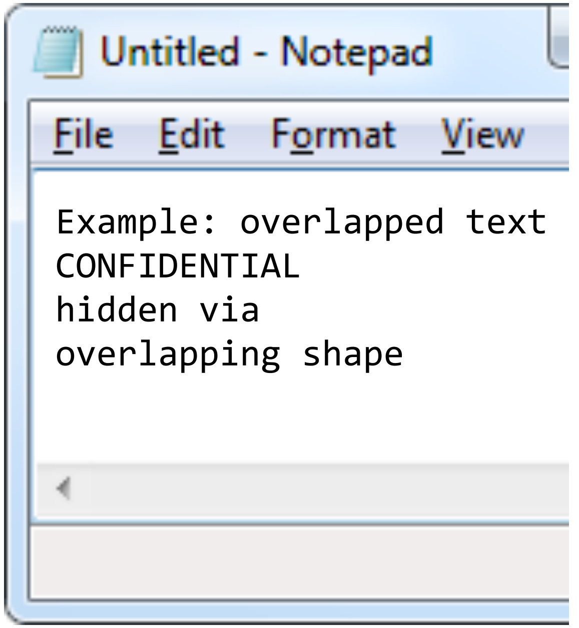
Try to get the secret from the next slide, with the same copy-paste trick...

Example: overlapped text

CONFIDENTIAL

Once again...

the text is behind the “CONFIDENTIAL” shape, but it’s still there!
the software selects everything
(not only the front layer)

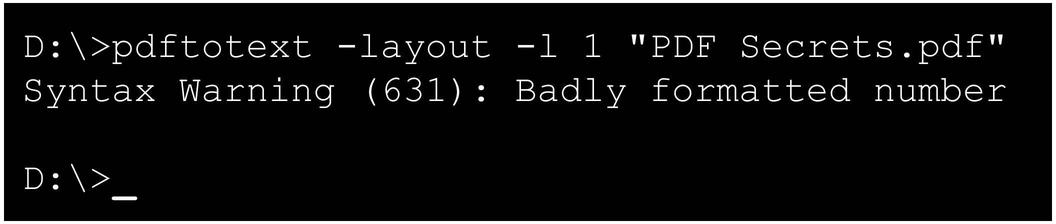


```
Example: overlapped text
CONFIDENTIAL
hidden via
overlapping shape
```

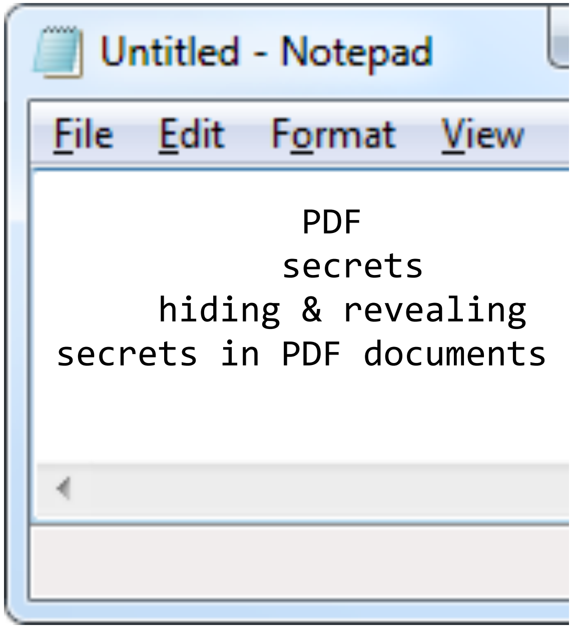
Better than “Select all”

`pdftotext` does it for you

instantly see which text is still hidden



```
D:\>pdftotext -layout -l 1 "PDF Secrets.pdf"
Syntax Warning (631): Badly formatted number

D:\>_
```

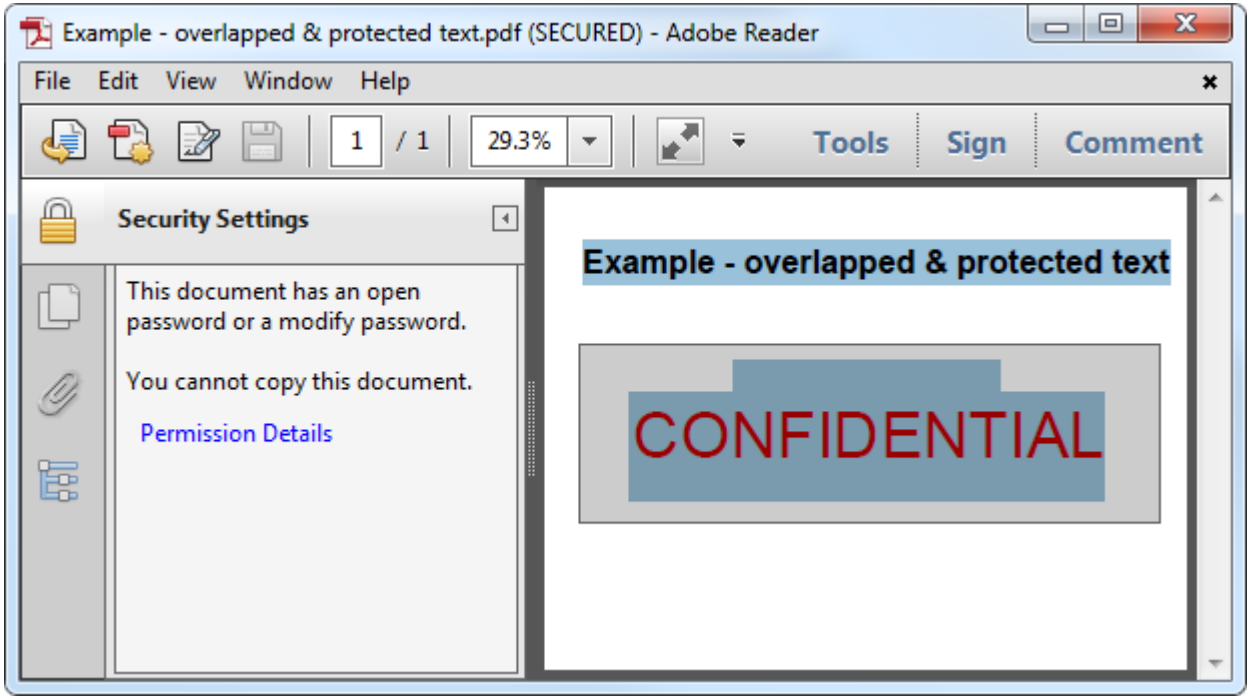


```
                PDF
              secrets
        hiding & revealing
secrets in PDF documents
```

But PDF can prevent that?

- yes, in theory
- but the text is still there, and decrypted

→ it can be circumvented



Example - overlapped & protected text.pdf (SECURED) - Adobe Reader

Security Settings

This document has an open password or a modify password.

You cannot copy this document.

Permission Details

Example - overlapped & protected text

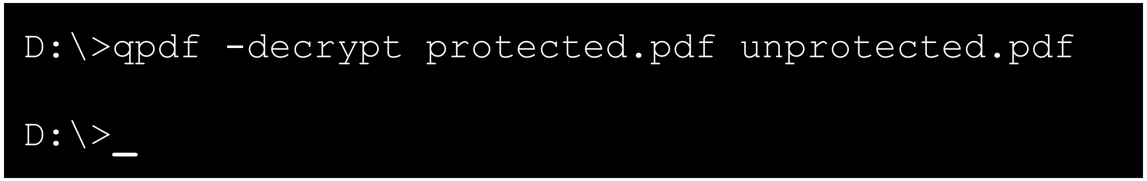
CONFIDENTIAL

Bypassing copy/paste protection

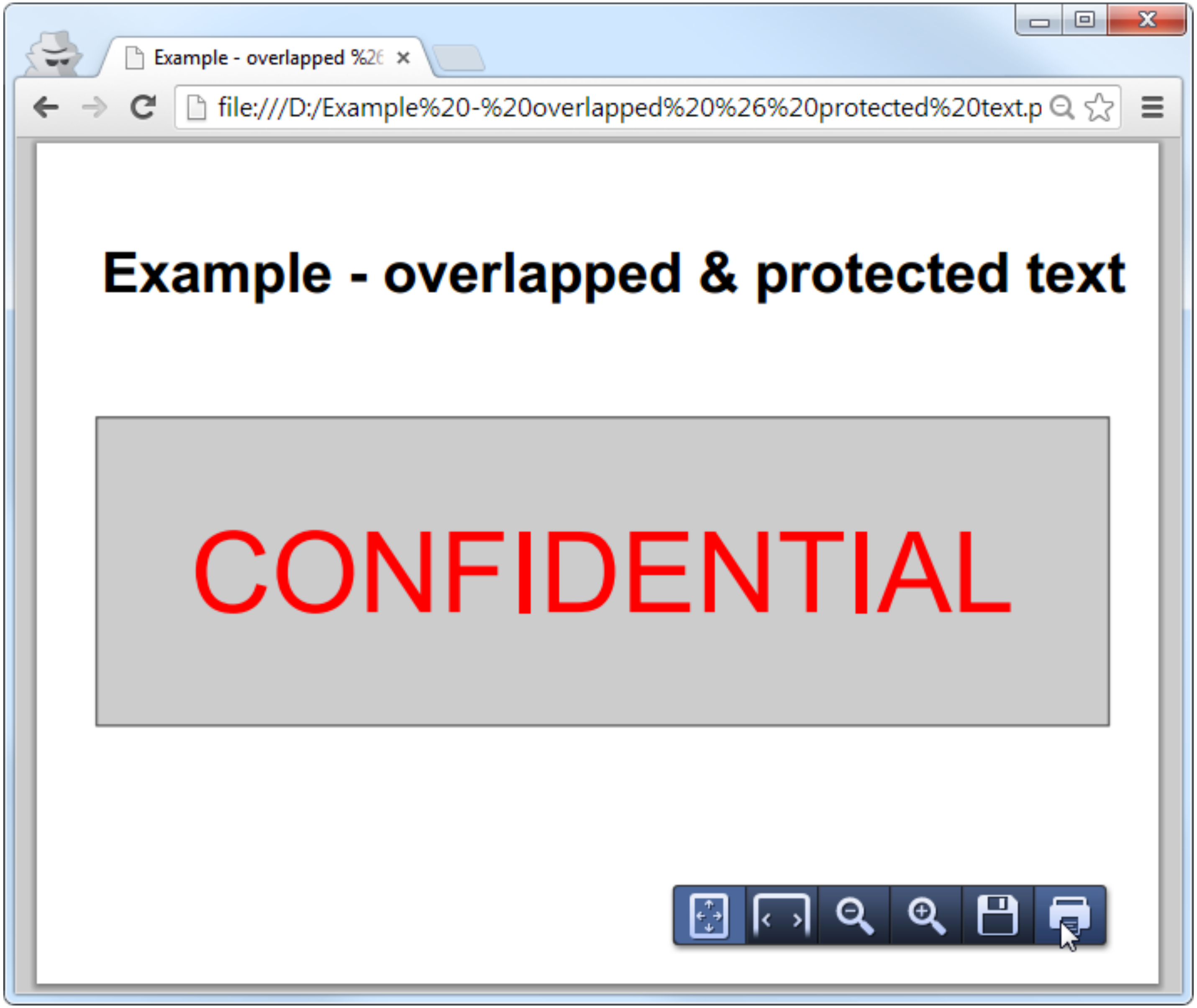
either:

- some readers just ignore it
    - like Evince
- generate a new file out of the original one
    - print PDF as PDF

      (not 100% compatible, but fast and usually works)
    - decrypt



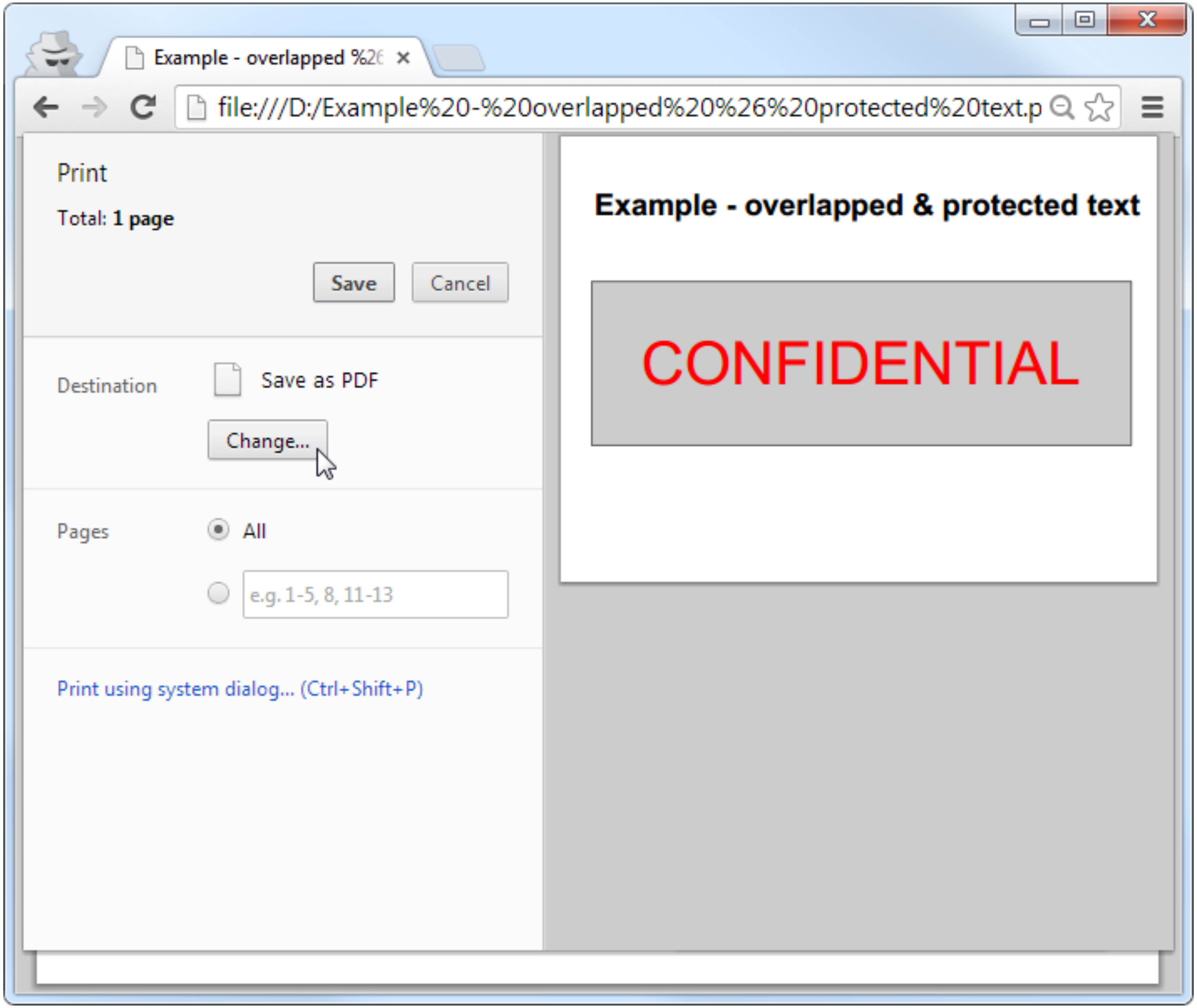
```
D:\>qpdf -decrypt protected.pdf unprotected.pdf

D:\>_
```


Example - overlapped %2€
file:///D:/Example%20-%20overlapped%20%26%20protected%20text.p

Example - overlapped & protected text

CONFIDENTIAL

1. open in chrome
2. print



file:///D:/Example%20-%20overlapped%20%26%20protected%20text.p

Print

Total: **1 page**

Destination Save as PDF

Pages ● All

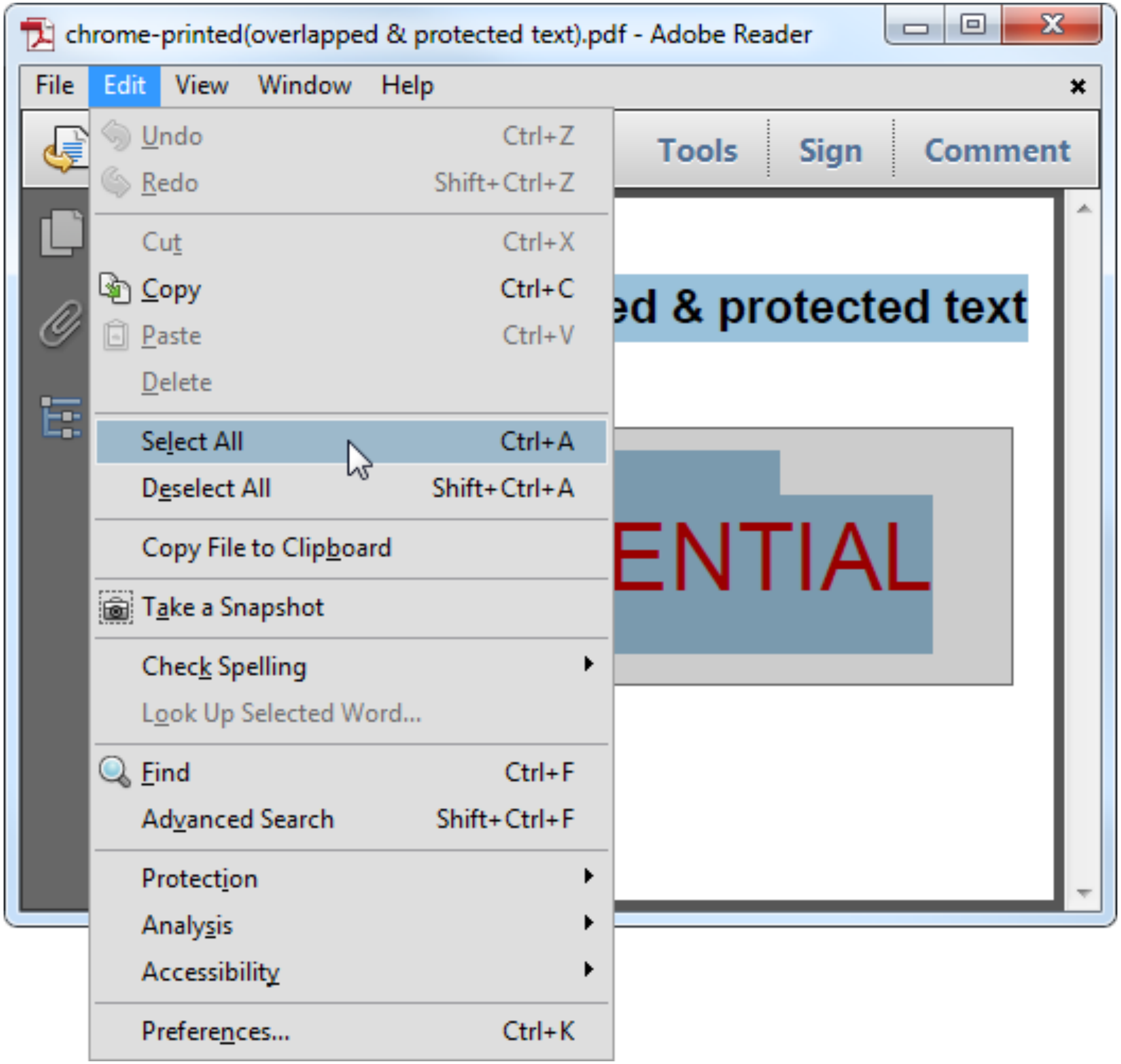
○ e.g. 1-5, 8, 11-13

Print using system dialog... (Ctrl+Shift+P)

Example - overlapped & protected text

CONFIDENTIAL

1. change printer as “Save as PDF”
2. Save



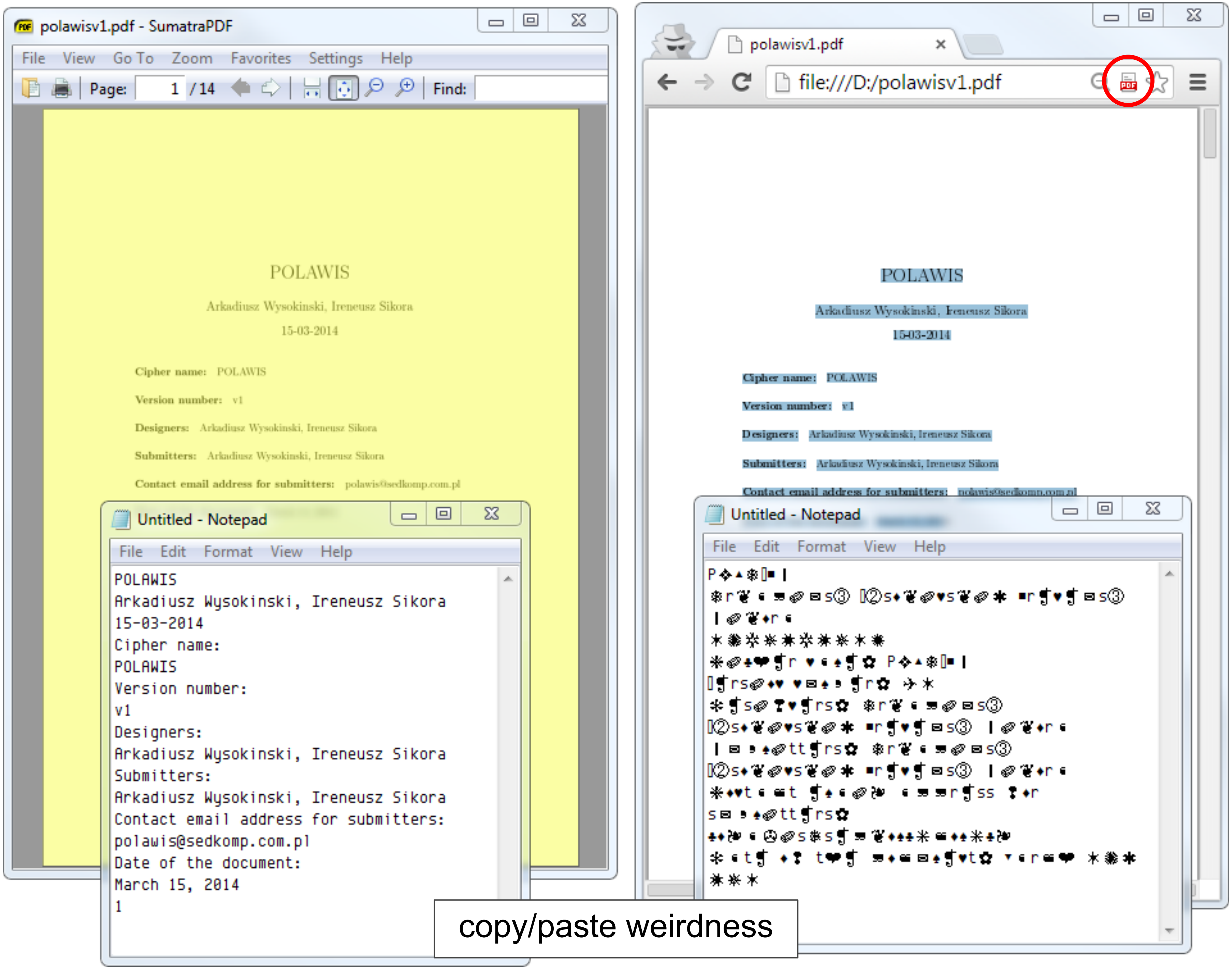
final document looks identical
not (SECURED) anymore

Copy/paste corruption

- sometimes, text can be copied,
  but it comes as corrupted
- it’s not protection, just incompatibility

→ try with another reader

- it could be abused
  - but it’s not easy to implement
  - and it’s still easy to recover content
    (it’s just a substitution cipher)


polawisv1.pdf - SumatraPDF
File View Go To Zoom Favorites Settings Help
Page: 1 / 14
Find:
polawisv1.pdf
file:///D:/polawisv1.pdf
Untitled - Notepad
File Edit Format View Help
Untitled - Notepad
File Edit Format View Help

POLAWIS

Arkadiusz Wysokinski, Ireneusz Sikora

15-03-2014

Cipher name: POLAWIS

Version number: v1

Designers: Arkadiusz Wysokinski, Ireneusz Sikora

Submitters: Arkadiusz Wysokinski, Ireneusz Sikora

Contact email address for submitters: polawis@sedkomp.com.pl

```
POLAWIS
Arkadiusz Wysokinski, Ireneusz Sikora
15-03-2014
Cipher name:
POLAWIS
Version number:
v1
Designers:
Arkadiusz Wysokinski, Ireneusz Sikora
Submitters:
Arkadiusz Wysokinski, Ireneusz Sikora
Contact email address for submitters:
polawis@sedkomp.com.pl
Date of the document:
March 15, 2014
1
```

copy/paste weirdness

Ok, a last one

is it hopeless?

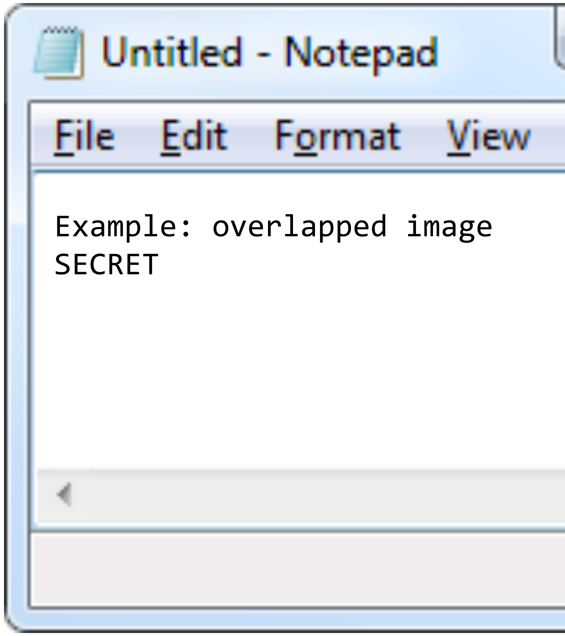
try this one...

Example: overlapped image

SECRET

Failure?

the secret behind the shape is a picture:
→ it’s not copied as text by standard software
(common softwares don’t copy pictures)



Example: overlapped image
SECRET

Does it means we’re safe?

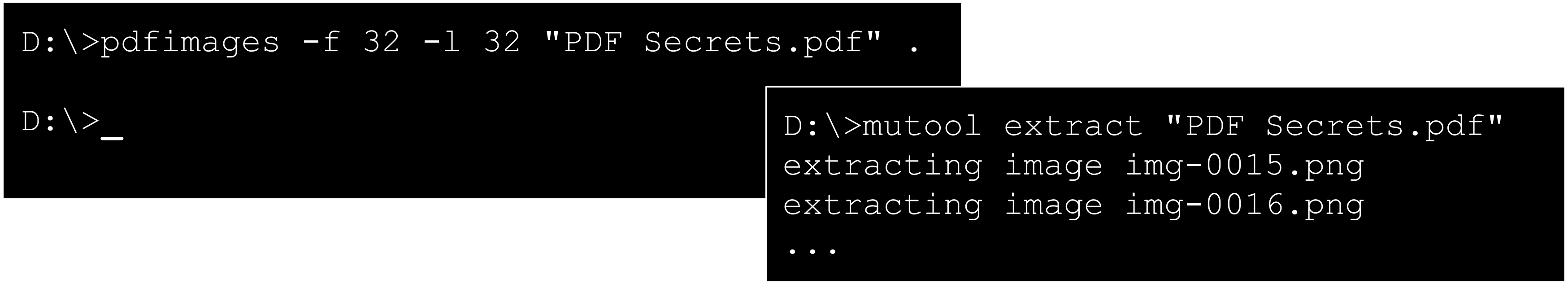
No:

the image is still present in the PDF document.

→ it’s trivial to extract it with a standard tool

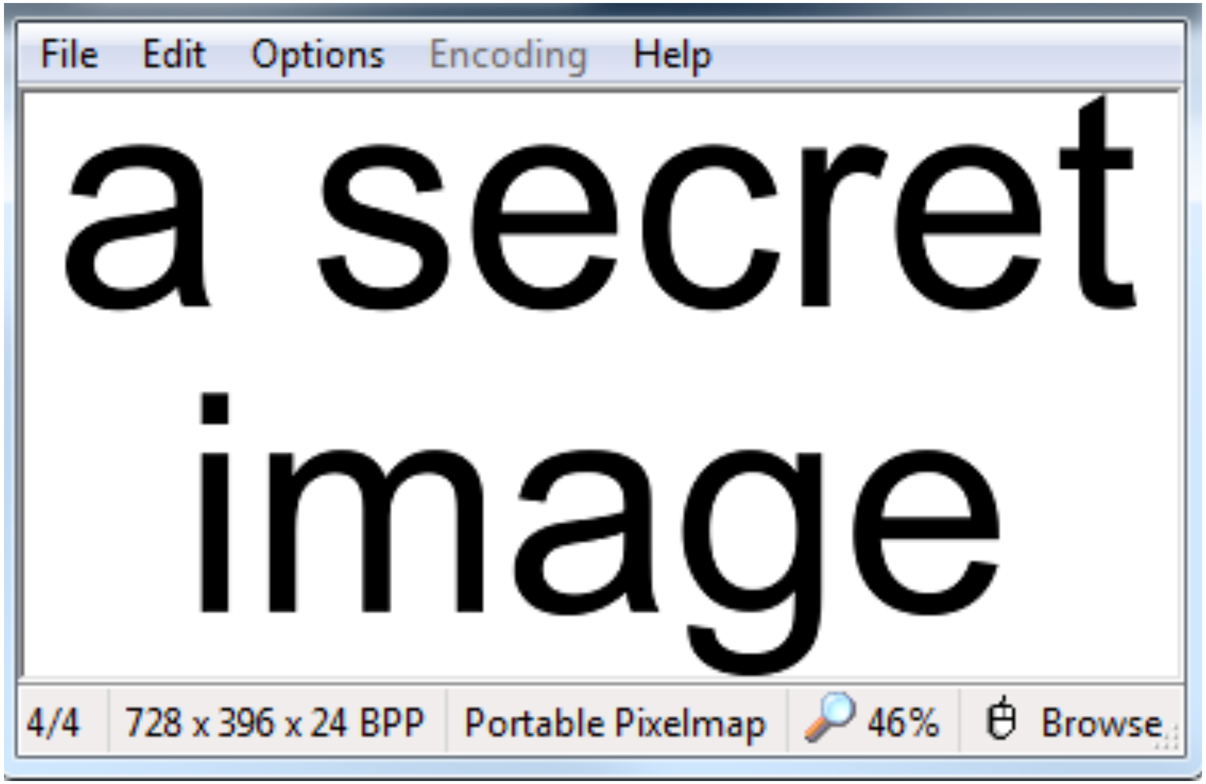
Example:

use `PDFImages` (or `mutool`)



```
D:\>pdfimages -f 32 -l 32 "PDF Secrets.pdf" .

D:\>_
```

```
D:\>mutool extract "PDF Secrets.pdf"
extracting image img-0015.png
extracting image img-0016.png
...
```



extracting our secret image directly from the file

Conclusion

on Part I / III

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

but there are more advanced tricks!

→ need to study PDF internals

PDF 101

basics of the PDF file format

Part II / III

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
      >>
    >>
  >>
>>
endobj

4 0 obj
<< /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`

<OBJECT NUMBER> <REVISION NUMBER> R

```
[...]

3 0 obj
value
endobj
```

BINARY STREAMS

BINARY STREAM ARE STORED IN SEPARATE OBJECTS LIKE THIS:

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

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

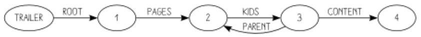
`xref`
<STARTING OBJECT> <OBJECT COUNT>
FOLLOWED BY XREF ENTRIES:
IF (OBJECT IN USE)
  <OFFSET:10> <GENERATION:5> `n`
ELSE
  <NEXT_FREE_OBJECT:10> <GENERATION:5> `f`

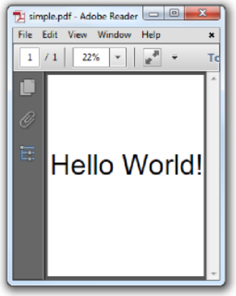
END OF THE FILE

`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





PDF101 an Adobe document walkthrough

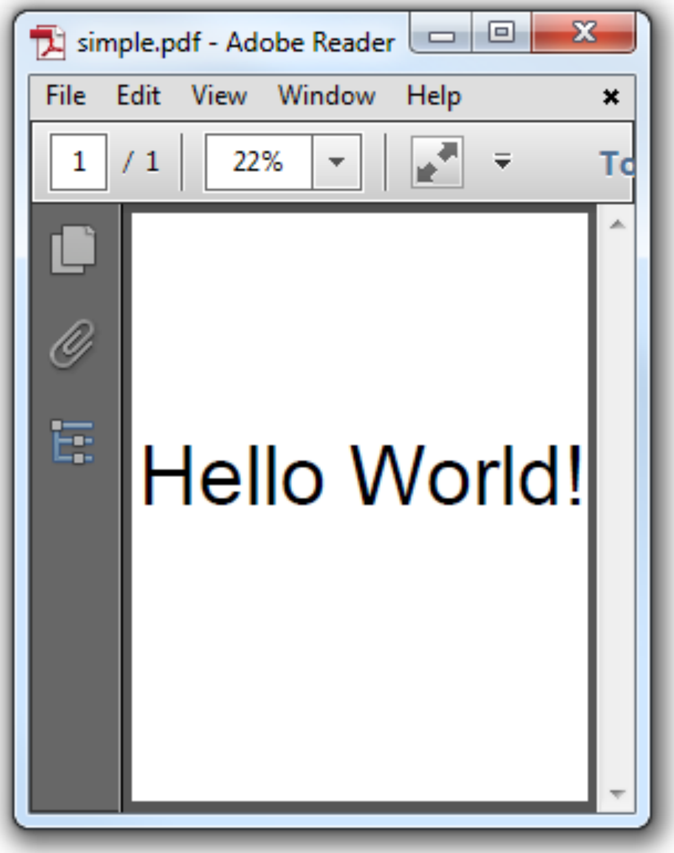
PORTABLE DOCUMENT FORMAT

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



Hello World!

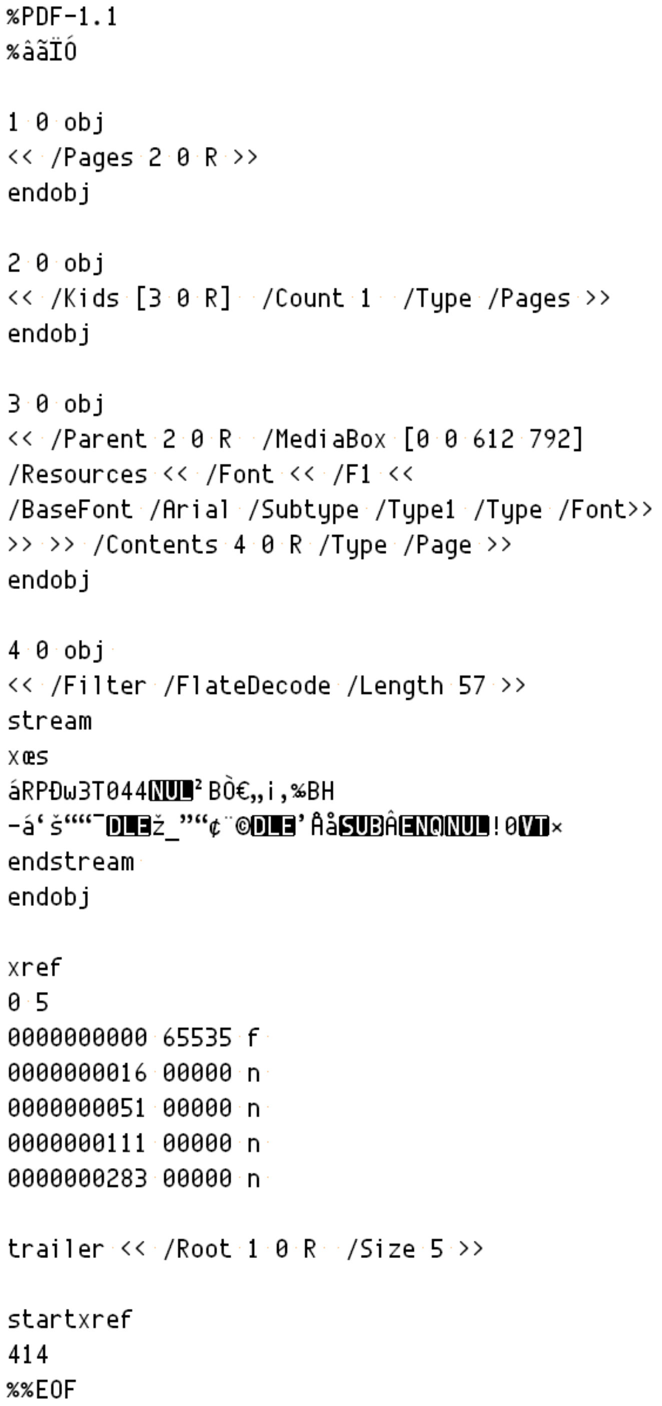
==

```
%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
áRPĐw3T044NUL²BÒ€„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 stream →

(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++

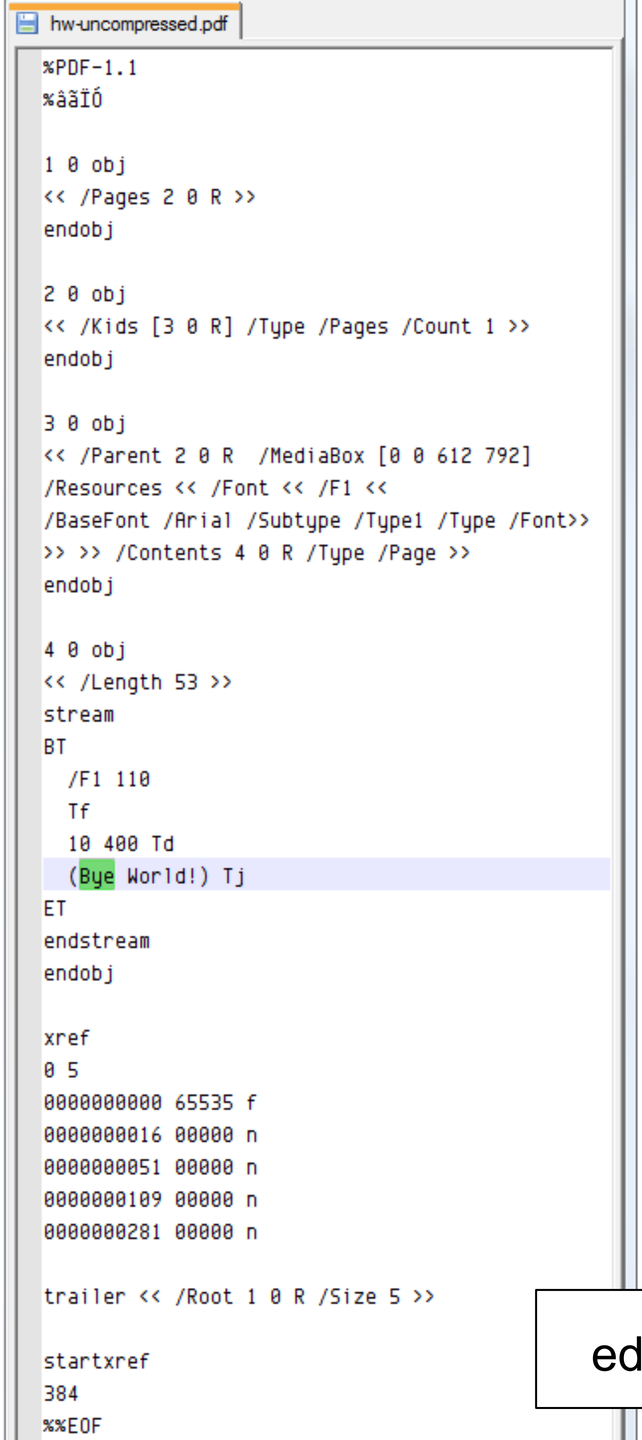
http://notepad-plus-plus.org/

Recommended environment

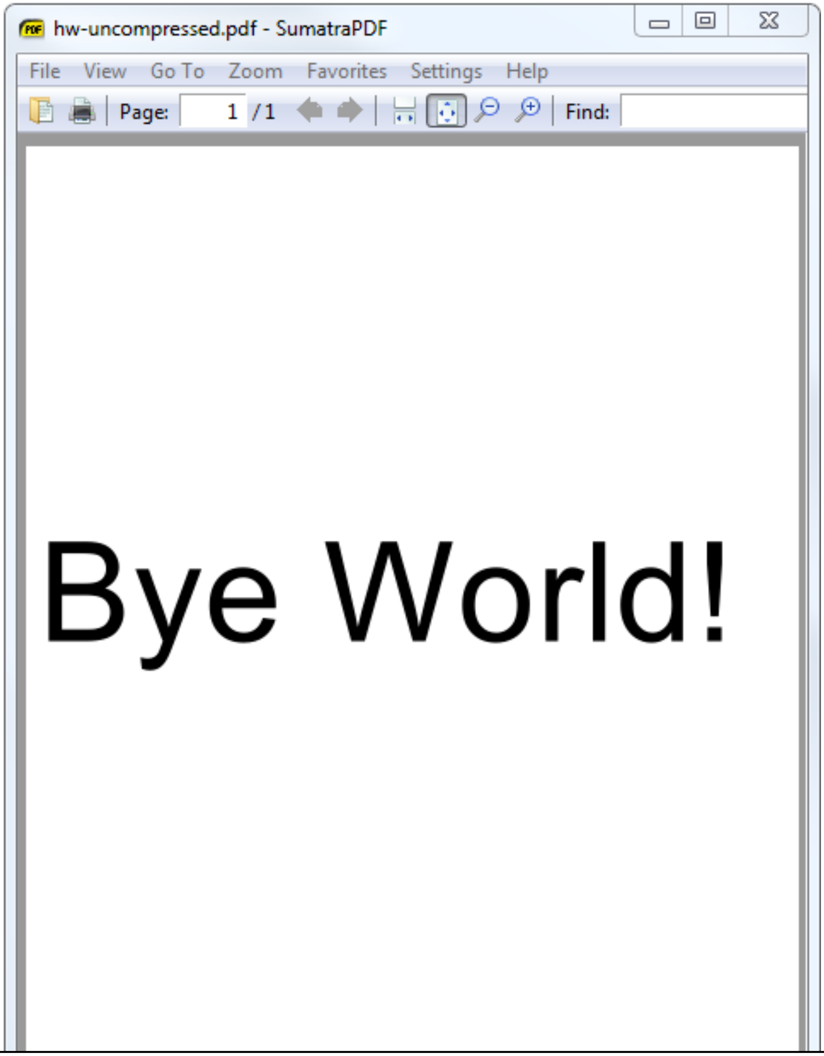
- text editor
- Sumatra
  - single-file viewer
  - updates on the fly

- a tool to decompress streams
  - (explanations later)

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



```
%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
  (Bye 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
```



Bye World!

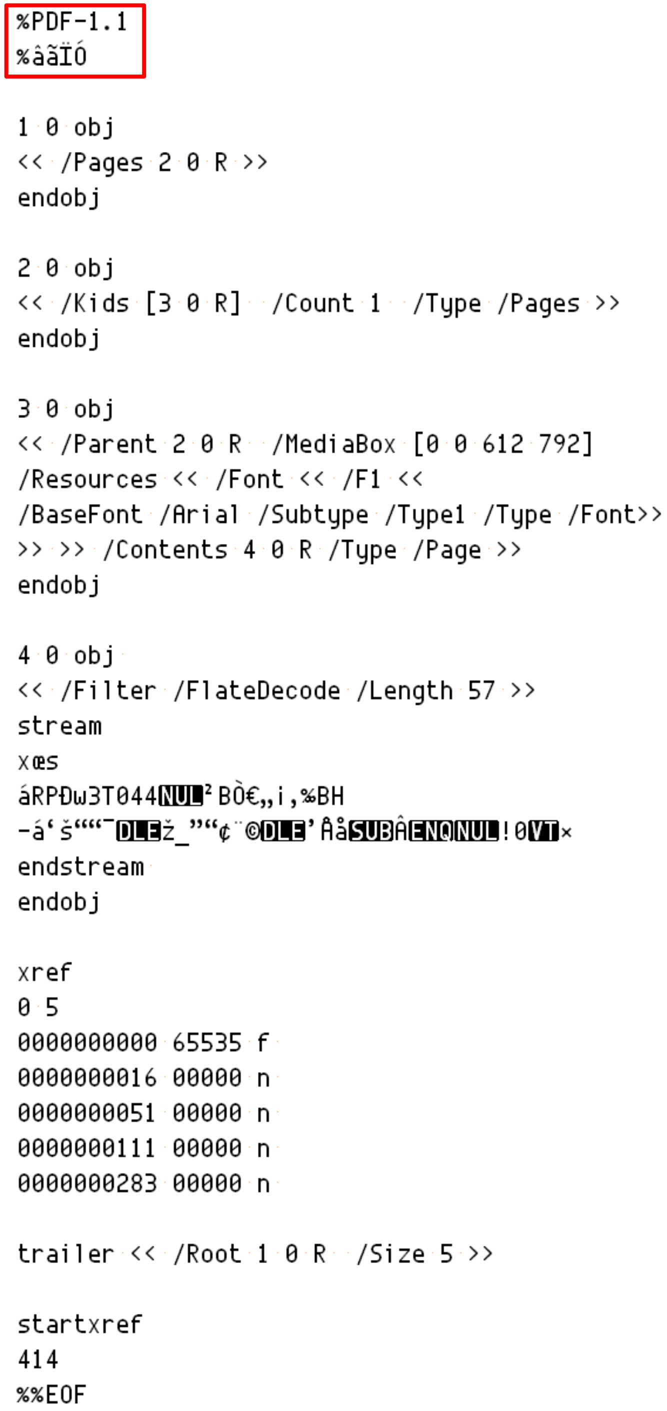
editing and viewing the changes on the fly

A PDF structure

1. header
   - signature
2. body
   - objects
3. cross-reference table
4. trailer
5. xref pointer
6. 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

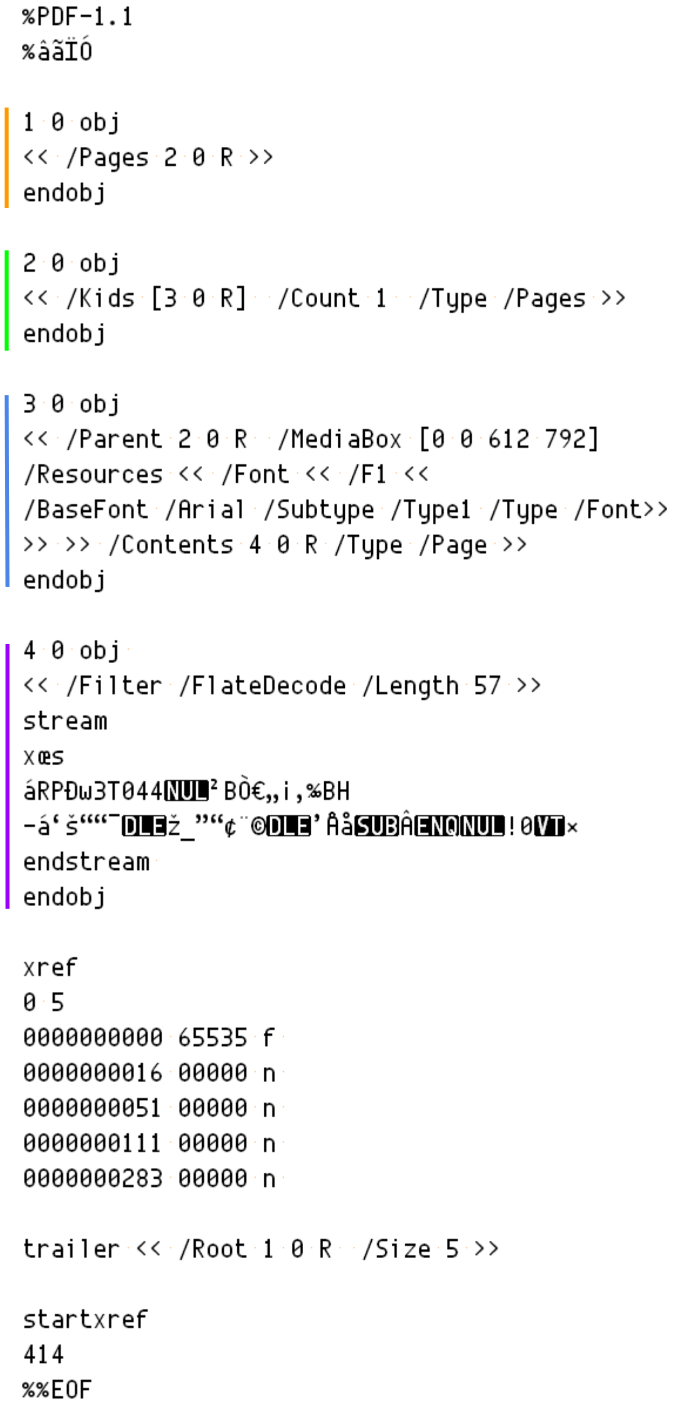


```
%PDF-1.1
%âãÏÓ

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



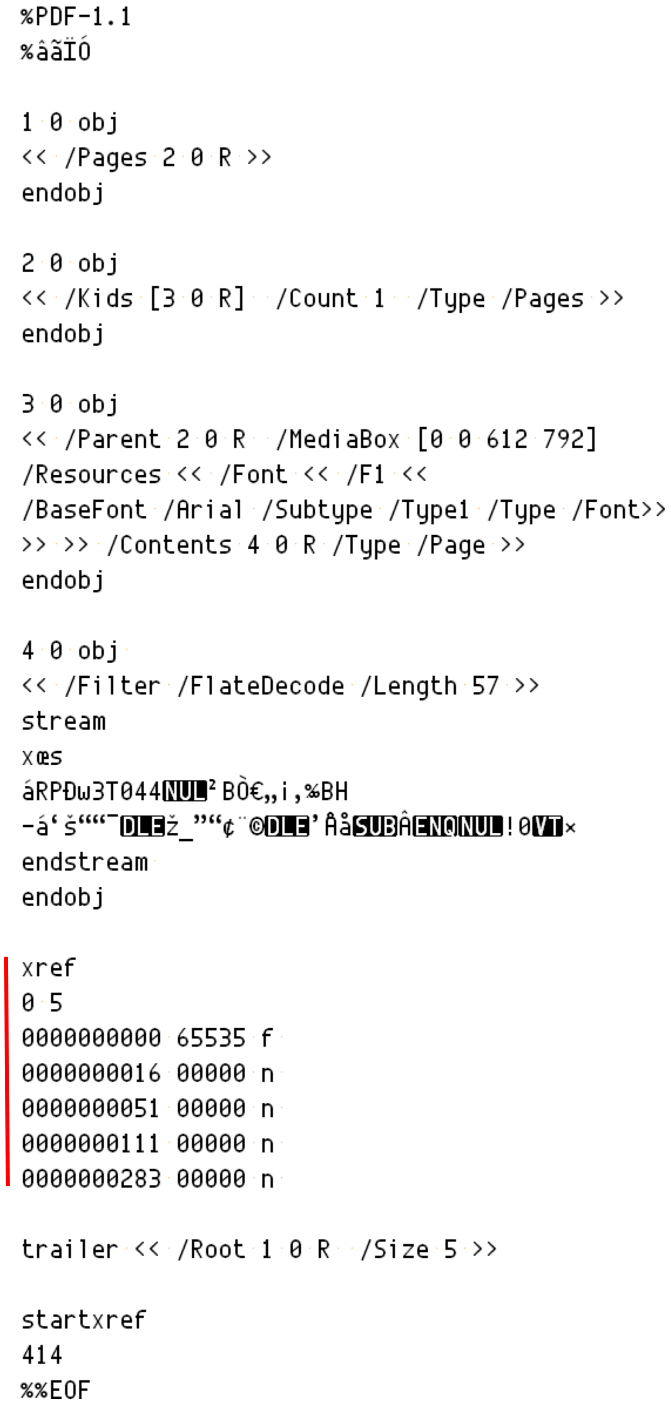
```
%PDF-1.1
%âãÏÓ

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

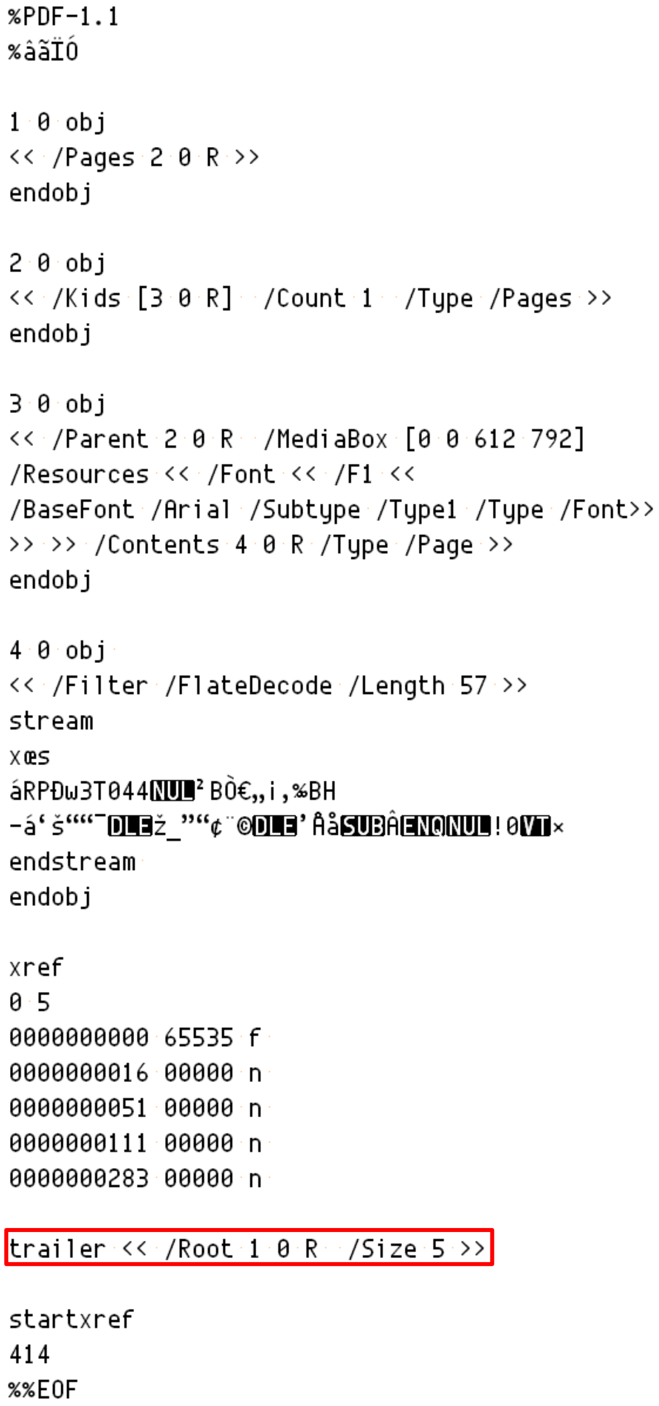


```
%PDF-1.1
%âãÏÓ

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
áRPĐw3T044NUL²BÒ€„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. object-like content

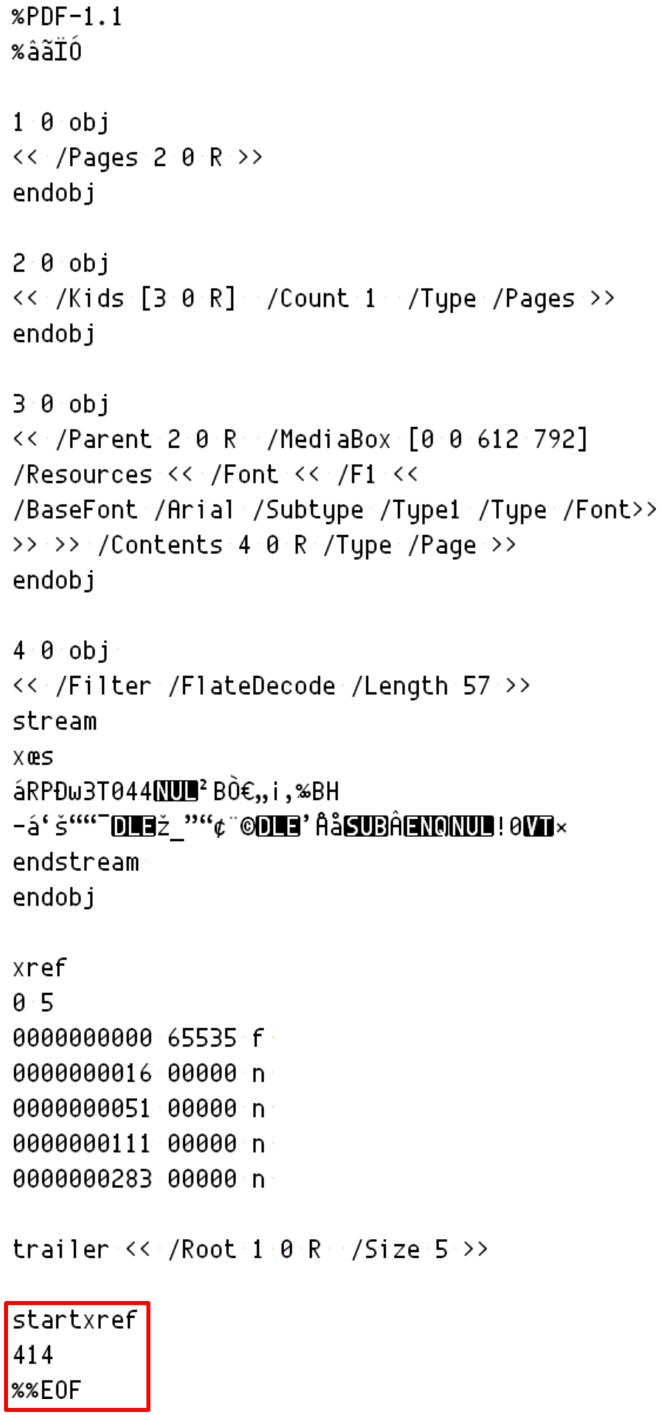
- defines the “root” object
    - `/Size` = #(xref elements)



```
%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
áRPĐw3T044NUL²BÒ€„i,%BH
-á‘š““‘¯DLEž_”“¢¨©DLE’ÂåSUBÂENQNUL!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
áRPĐw3T044NUL²BÒ€„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
```

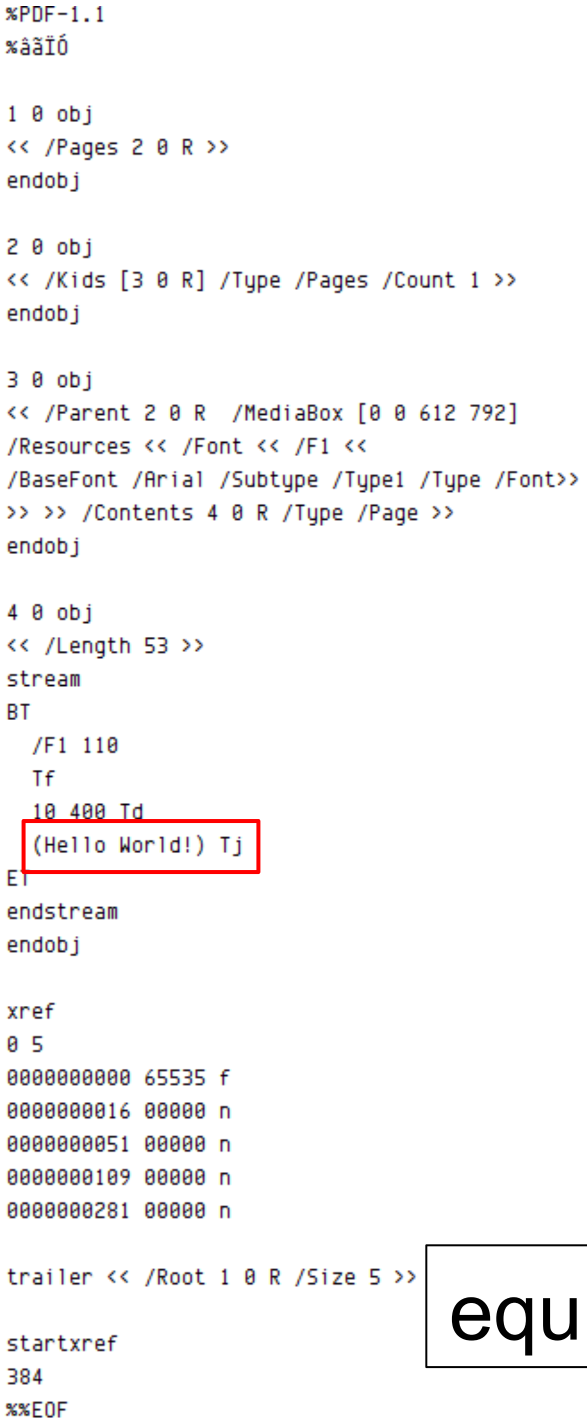
Basic types

names, strings, dictionaries...

Literals

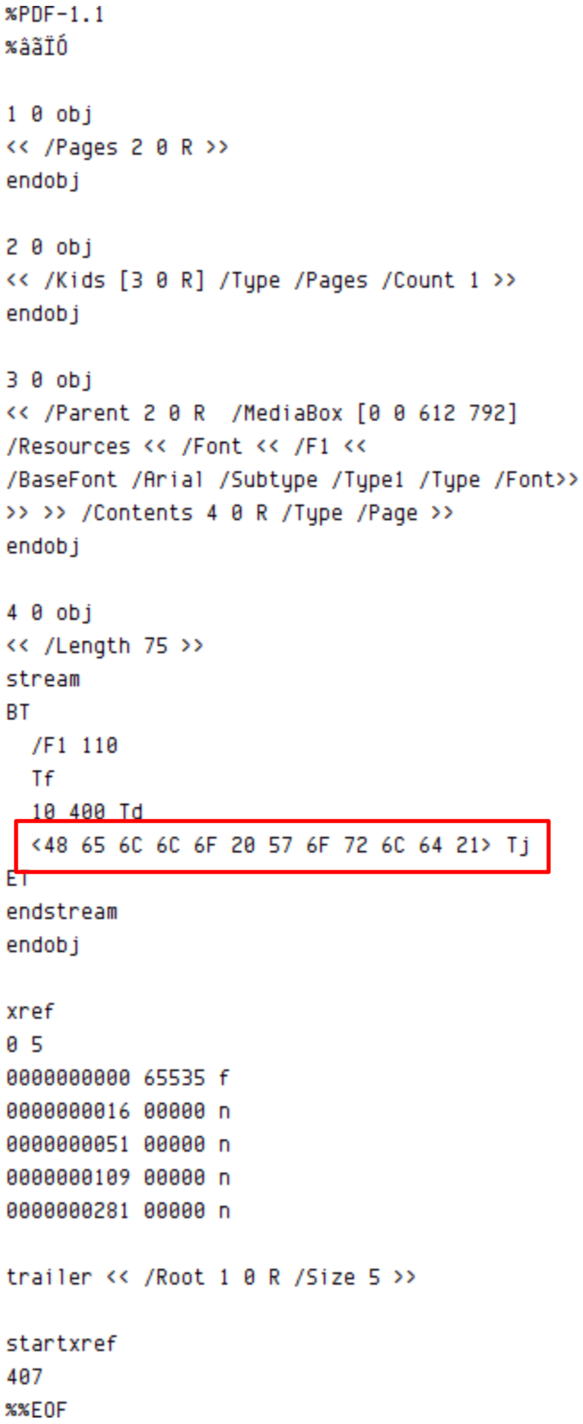
- **(**`string`**)**
- **<**`hex`**>**
- **%**`comment until line return`

- 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

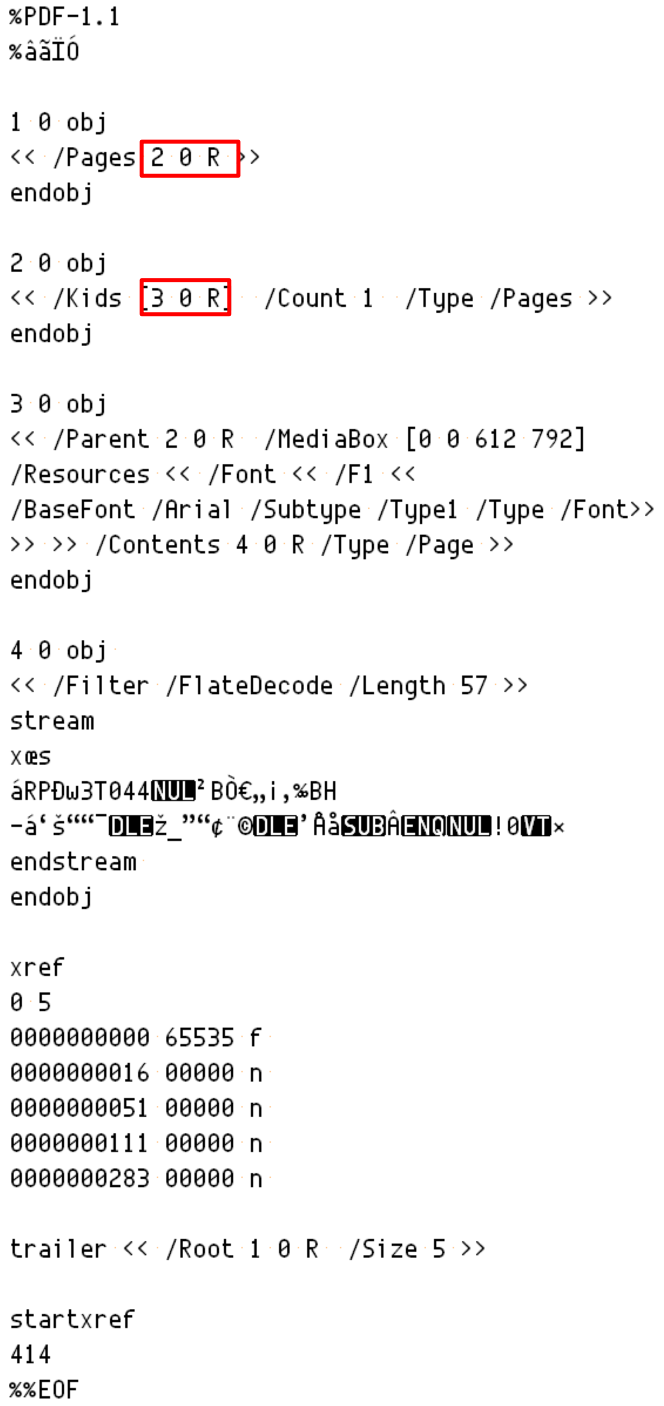
- `<object> <generation>` **R**

with

- 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
áRPĐw3T044NUL²BÒ€„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, but critical to understand

- `3 0 1` ⇒ 3 elements (3 numbers):
    1. 3
    2. 0
    3. 1
- `3 0 R` ⇒ 1 element:
    1. reference to “`3 0`”
        - object `3`
        - generation `0`

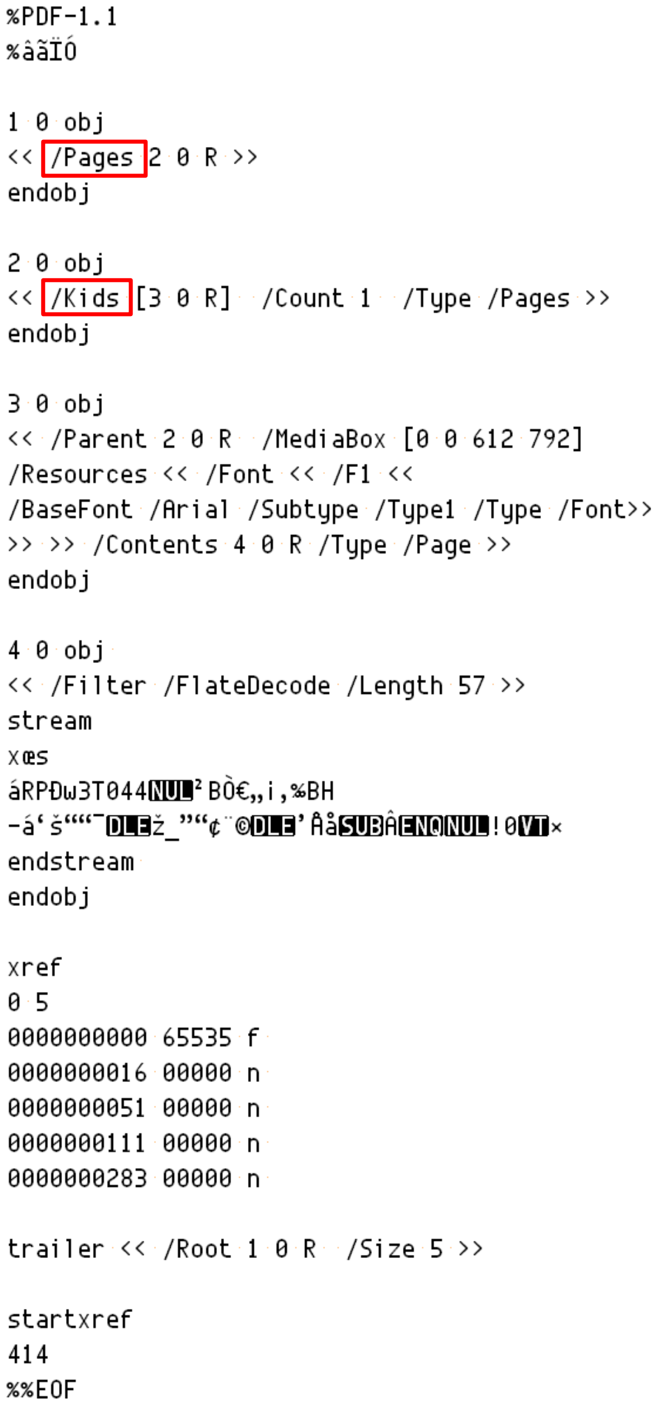
Other PDF syntax rules follow common-sense

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
áRPĐw3T044NUL²BÒ€„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
```

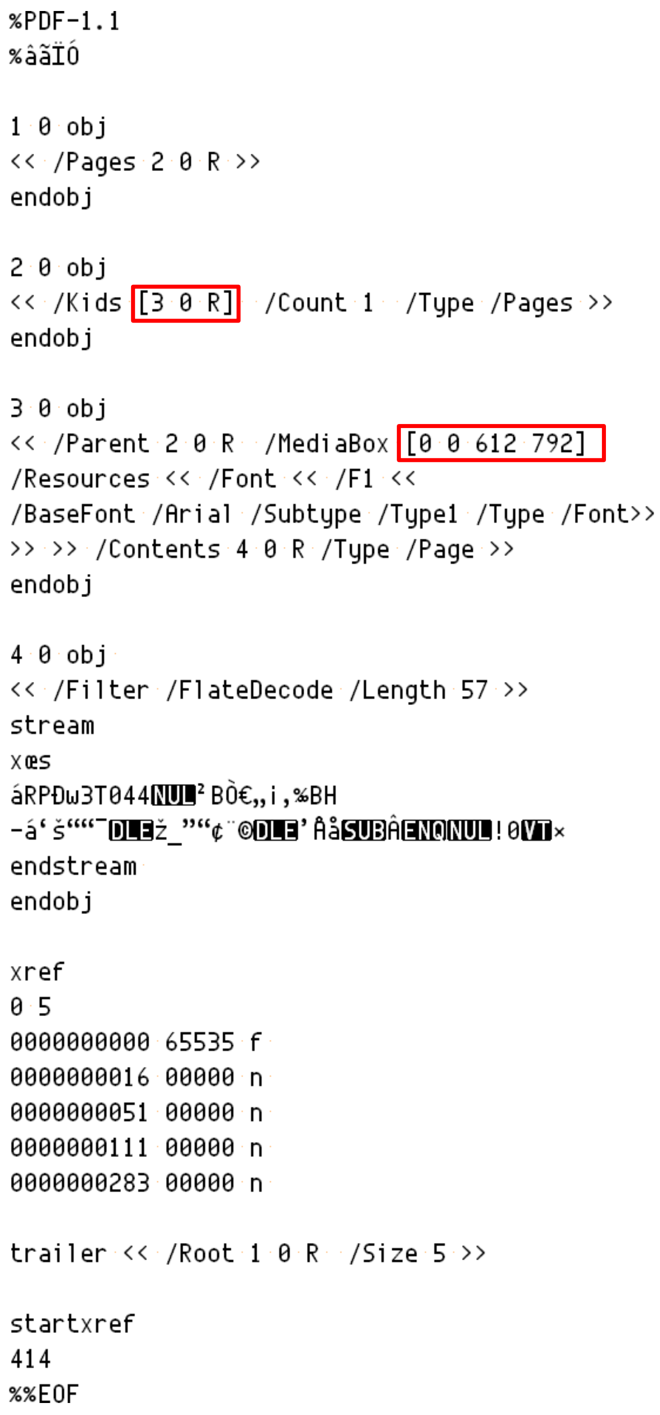
Array

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
áRPĐw3T044NUL²BÒ€„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:

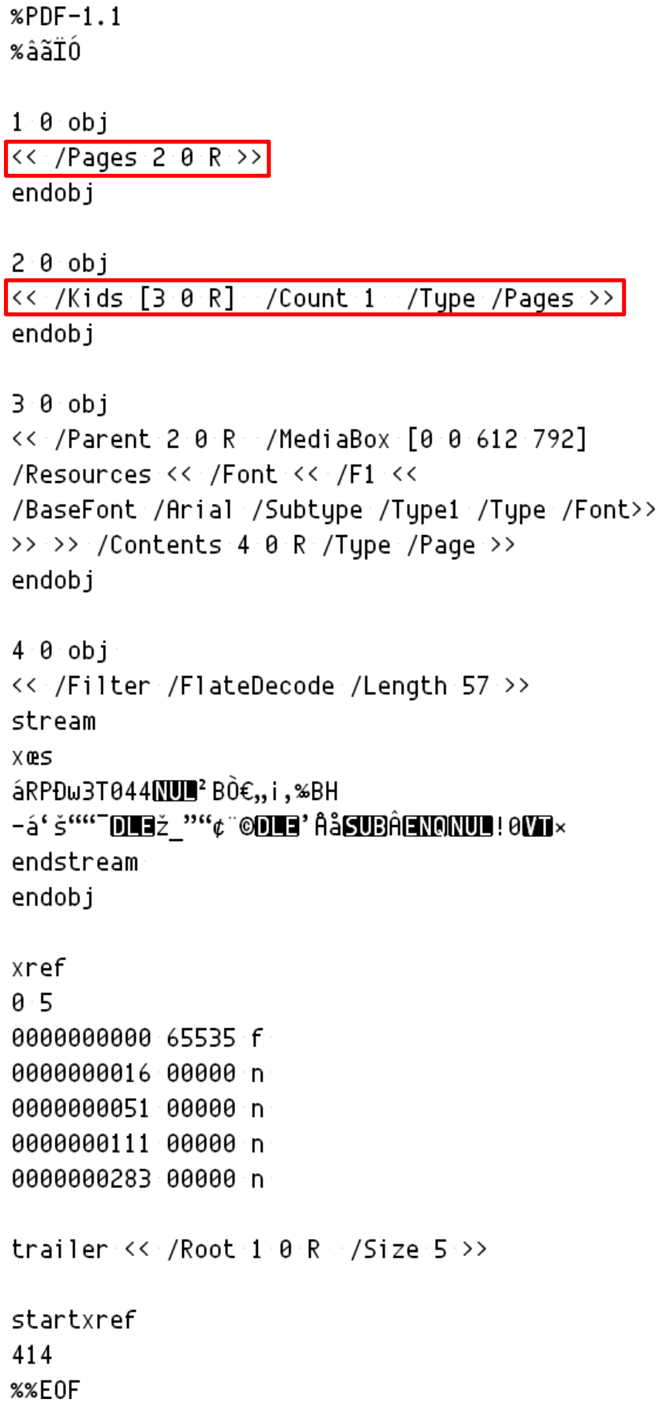
- **<<** [<name> <value>]* **>>**

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*



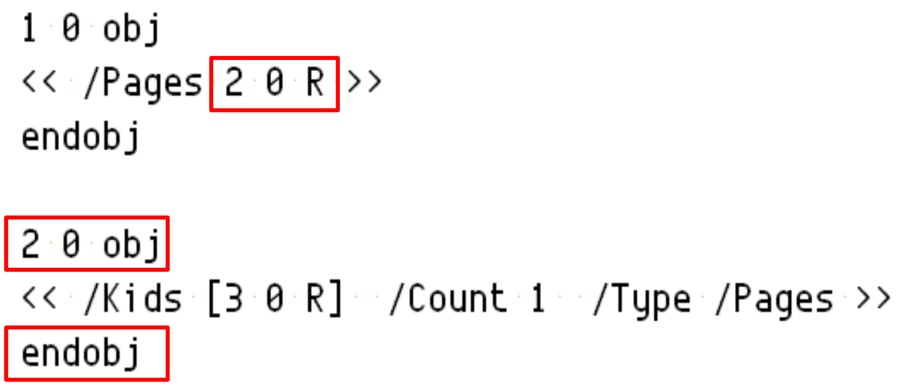
```
%PDF-1.1
%âãÏÓ

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
áRPĐw3T044NUL²BÒ€„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 replaced too...

Binary streams

parameters, filters...

Streams

syntax:

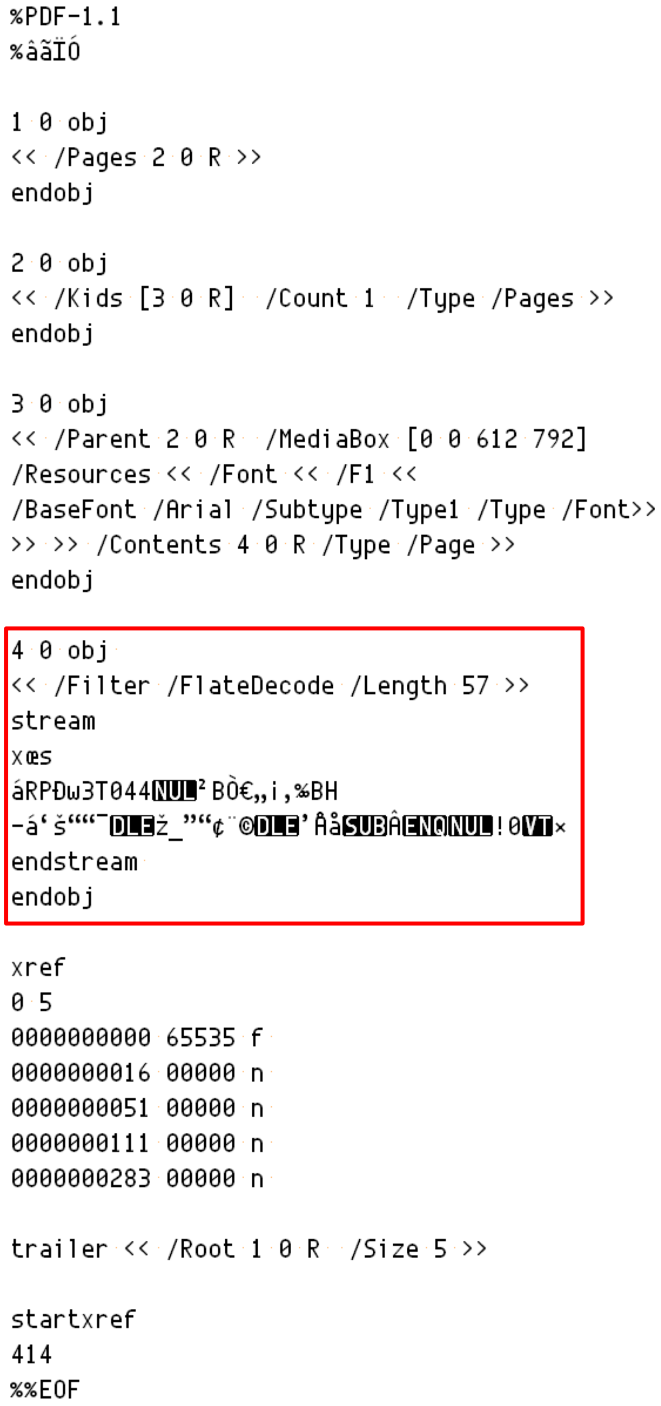
1. usual object declaration
2. parameters dictionary
3. `stream`

   `+ 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
```

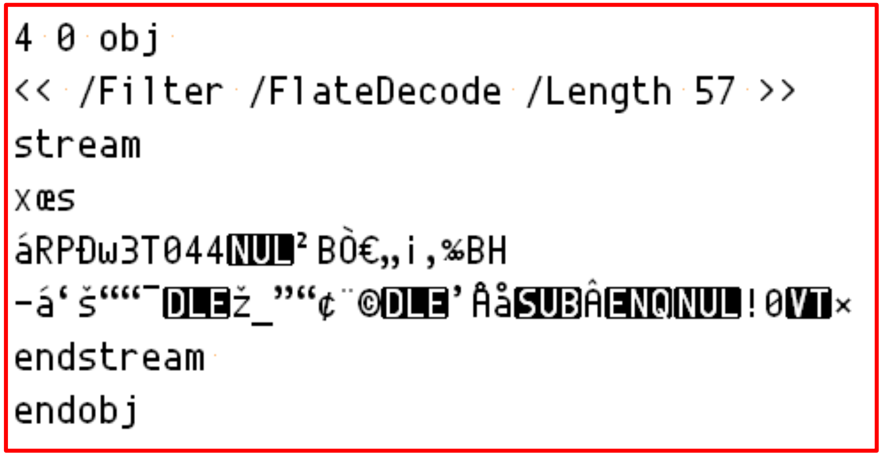
Example

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€„i,‰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
```

```
<< /Filter /FlateDecode

   /Length 57 >>
stream
xœs
áRPĐw3T044 ²BÒ€„¡,‰BH
-á‘š“““¯ž_”“¢¨©’ÅåÂ !0
×
endstream
```

these 2 streams are equivalent, just using a different encoding

```
<< /Filter
        [/ASCIIHexDecode
   /FlateDecode]
/Length 170 >>
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
```

```
<< /Filter /FlateDecode

   /Length 57 >>
stream
xœs
áRPÐw3T044 ²BÒ€„¡‚‰BH
-á‘š“““¯ž_”“¢¨©’ÅåÂ !0
×
endstream
```

/ASCIIHexDecode will
decode ASCII Hex to binary

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`

```
%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 4/7

4. `xref` gives offsets of each objects

```
%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 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

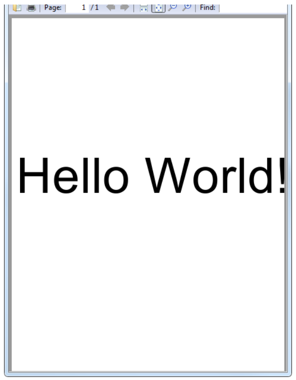
6. objects are parsed
   a. /Root object contains /Pages
   b. /Pages contains page array
      - /Kids
   c. each /Page has:
      - size: /MediaBox
      - /Contents
        - as stream object
      - /Resources
        - define /Font dictionary

```
%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



Hello World!

```
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
  - …

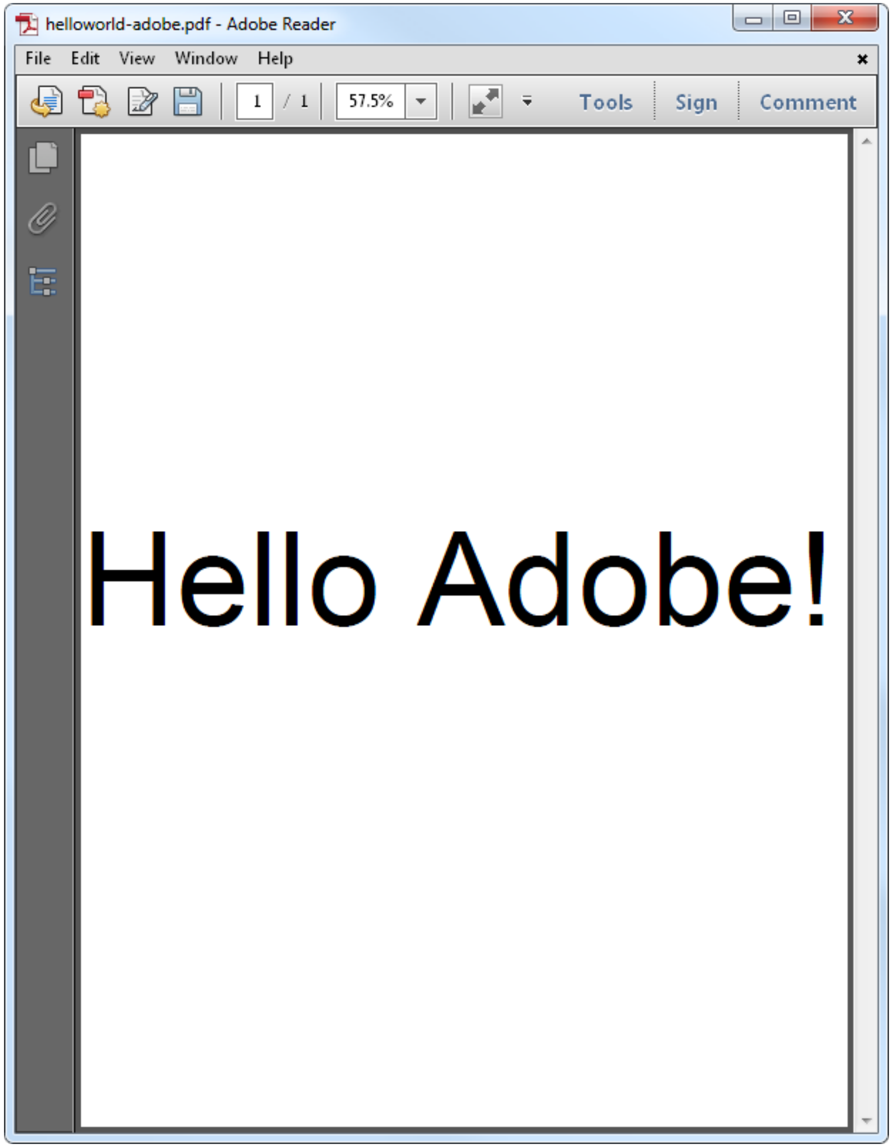
a generated *Hello World* typically weights >5 Kb

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>>>>
```


helloworld-adobe.pdf - Adobe Reader
File Edit View Window Help
1 / 1
57.5%
Tools
Sign
Comment

Hello Adobe!

a “Hello World” for Adobe, in 179 bytes

Conclusion

we’ve covered the basics of:

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

→ enough to play with PDF internals!

A technical perspective

Part III / III

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:

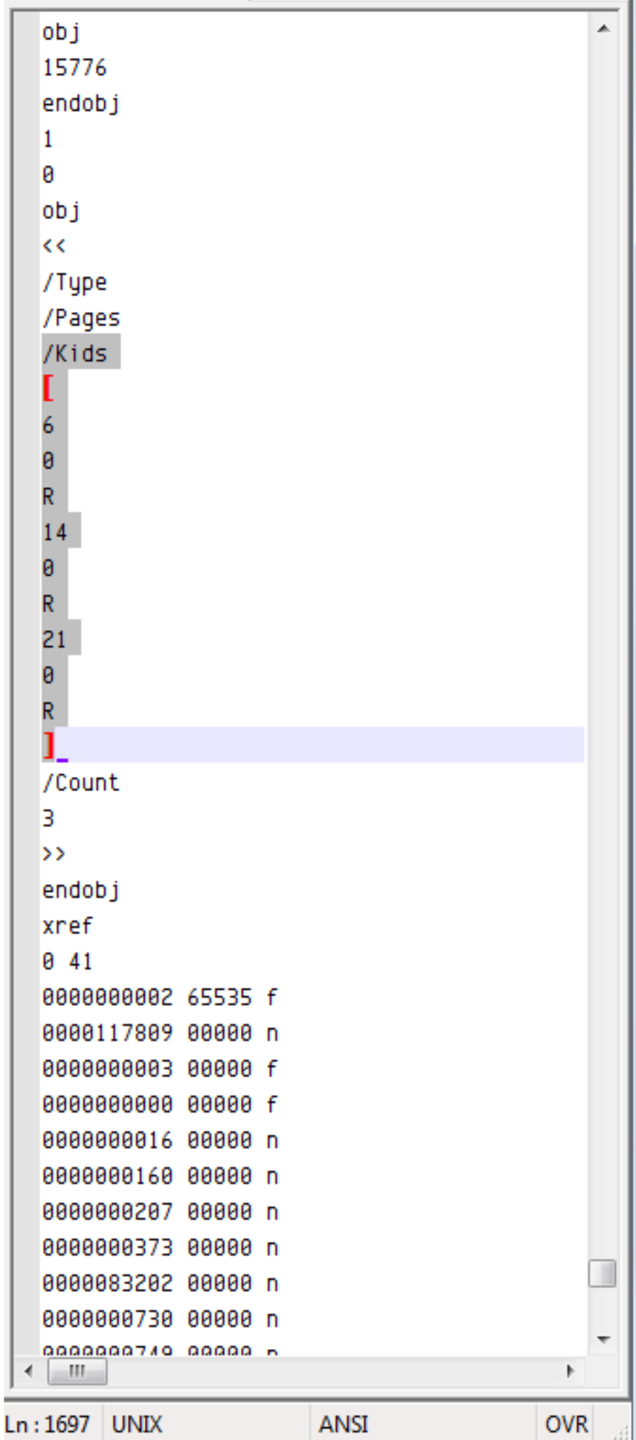
1. a stream object
2. as the `/Contents` of a `/Type /Page` object
3. in the `/Kids` array of a `/Type /Pages` object
4. as the value of `/Pages` in root object
5. as the value of `/Root` in the trailer

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

Remove a page ?

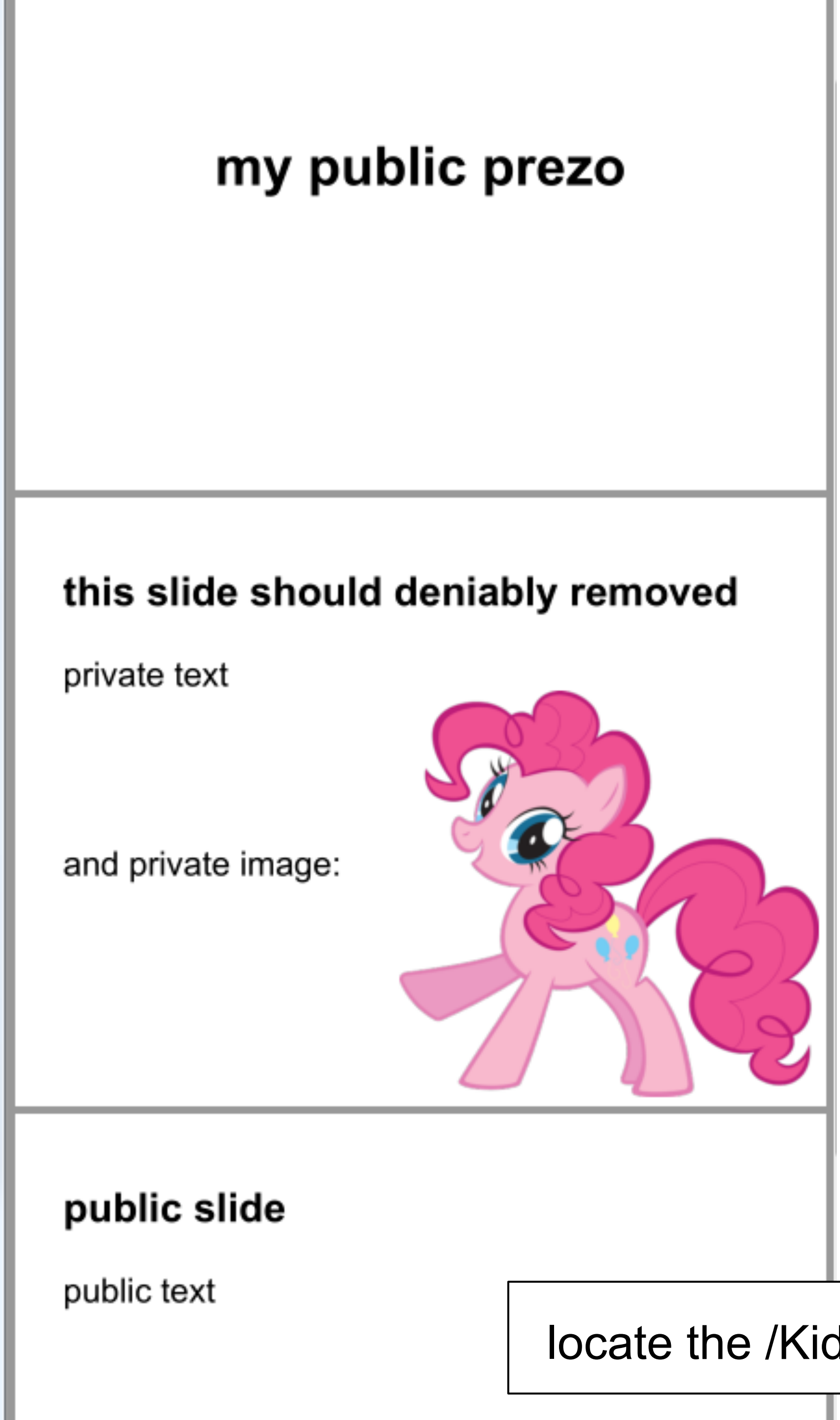
easy hiding

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
0000000749 00000 n
```

Ln:1697 UNIX ANSI OVR



my public prezo

this slide should deniably removed

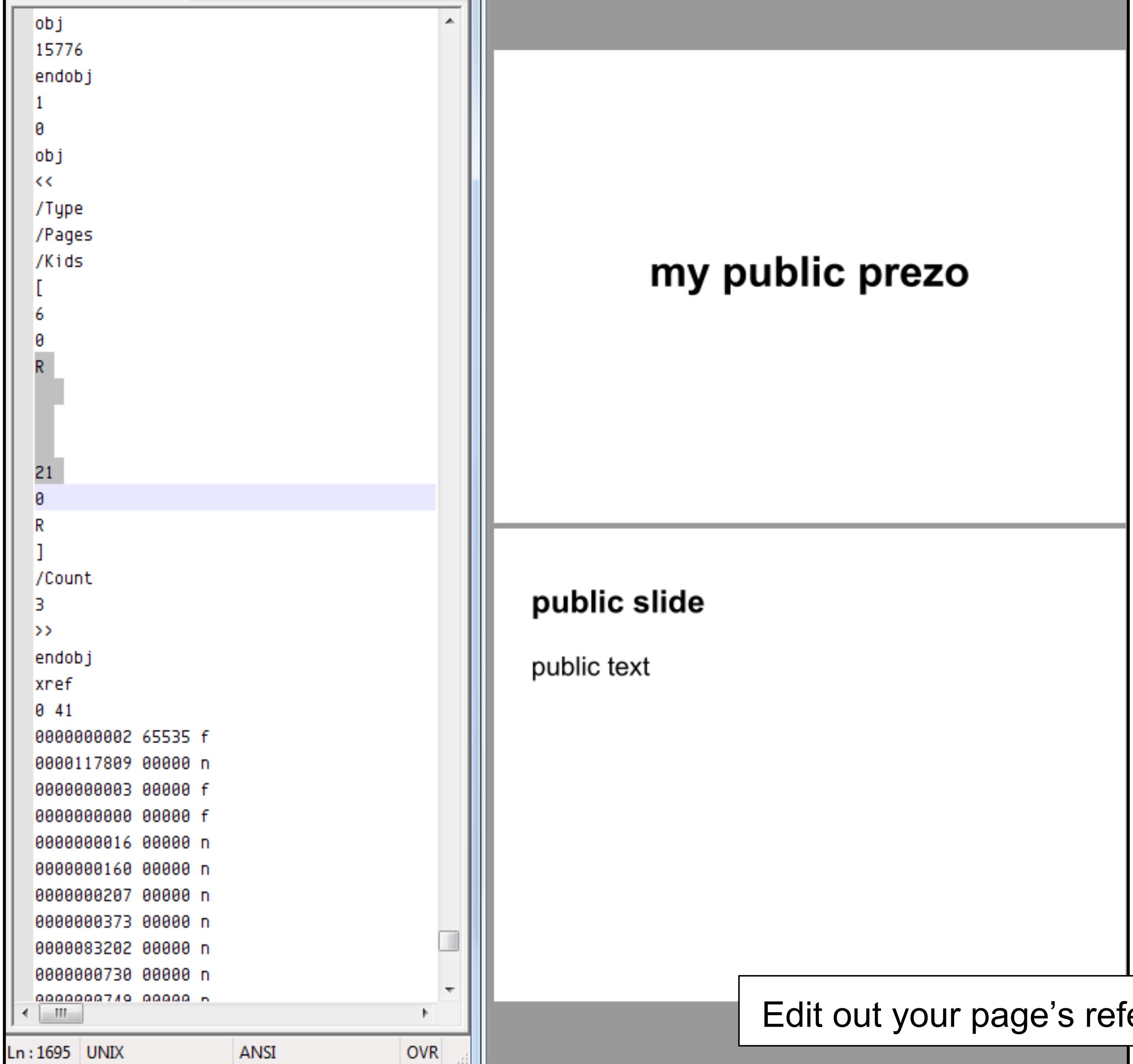
private text

and private image:

public slide

public text

locate the /Kids array



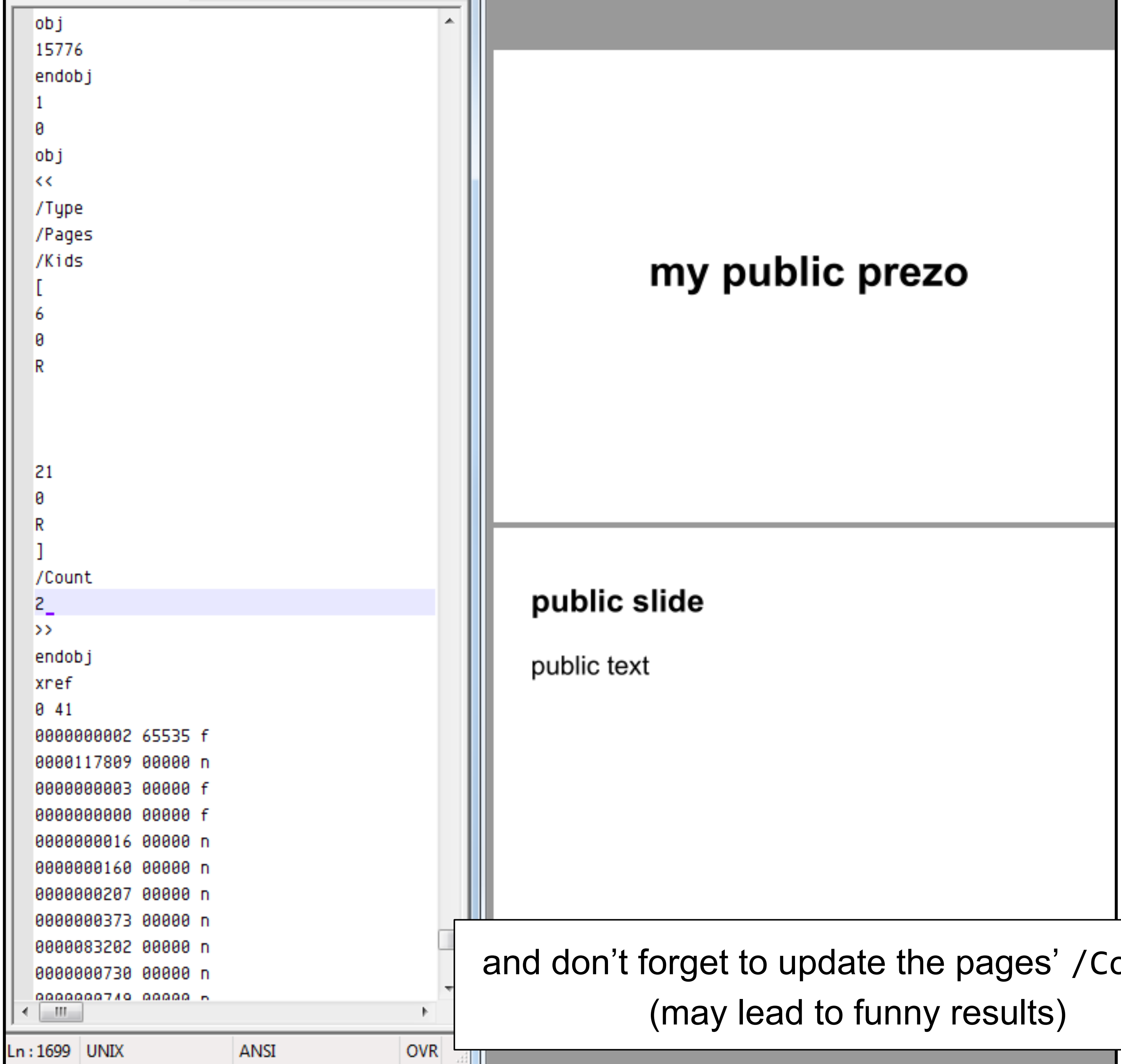
```
obj
15776
endobj
1
0
obj
<<
/Type
/Pages
/Kids
[
6
0
R



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
0000000749 00000 n
```

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
0000000749 00000 n
```

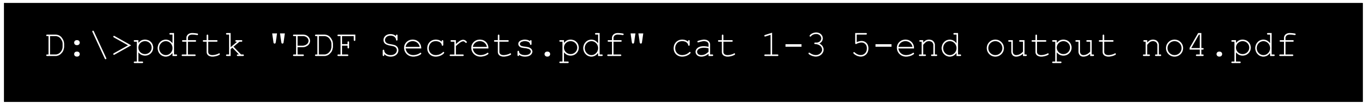
my public prezo

public slide

public text

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

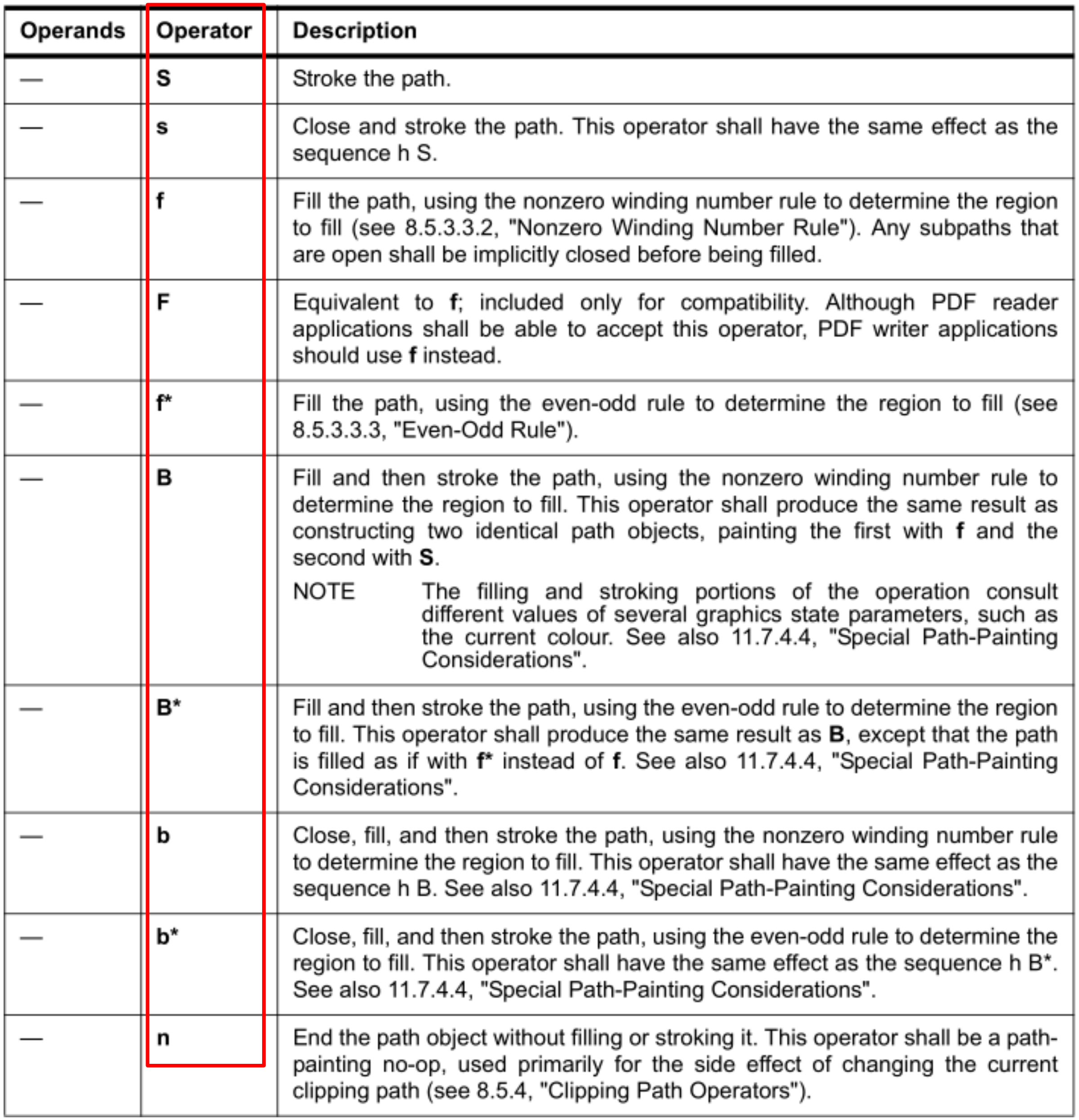
but the image contents (as objects) are still left!

and extractable!!

Erase overlapping element?

- remove paint/text operators from binary stream

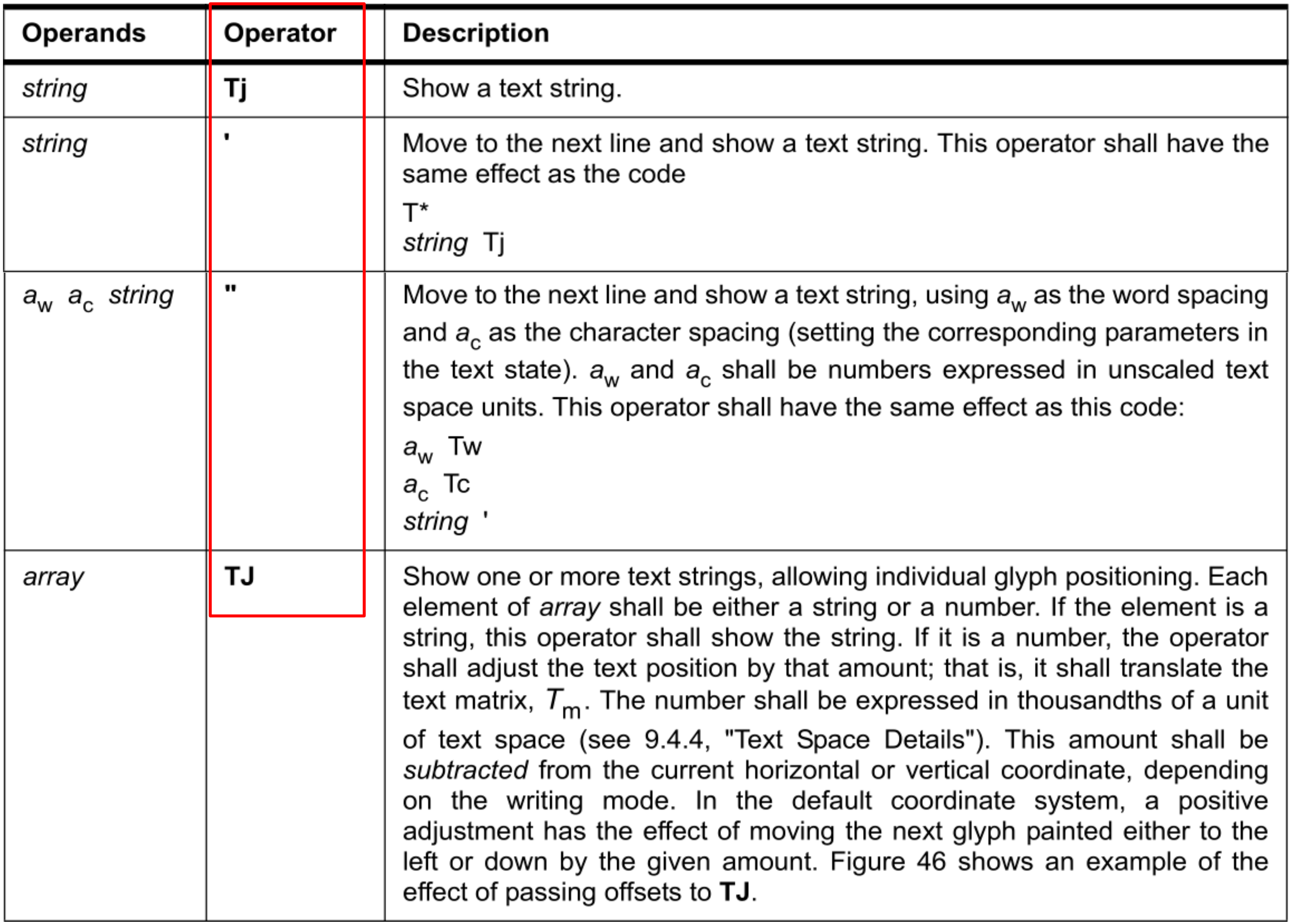
Hint:
overlapping elements might be
at the end of the stream,
as they were likely added last



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)

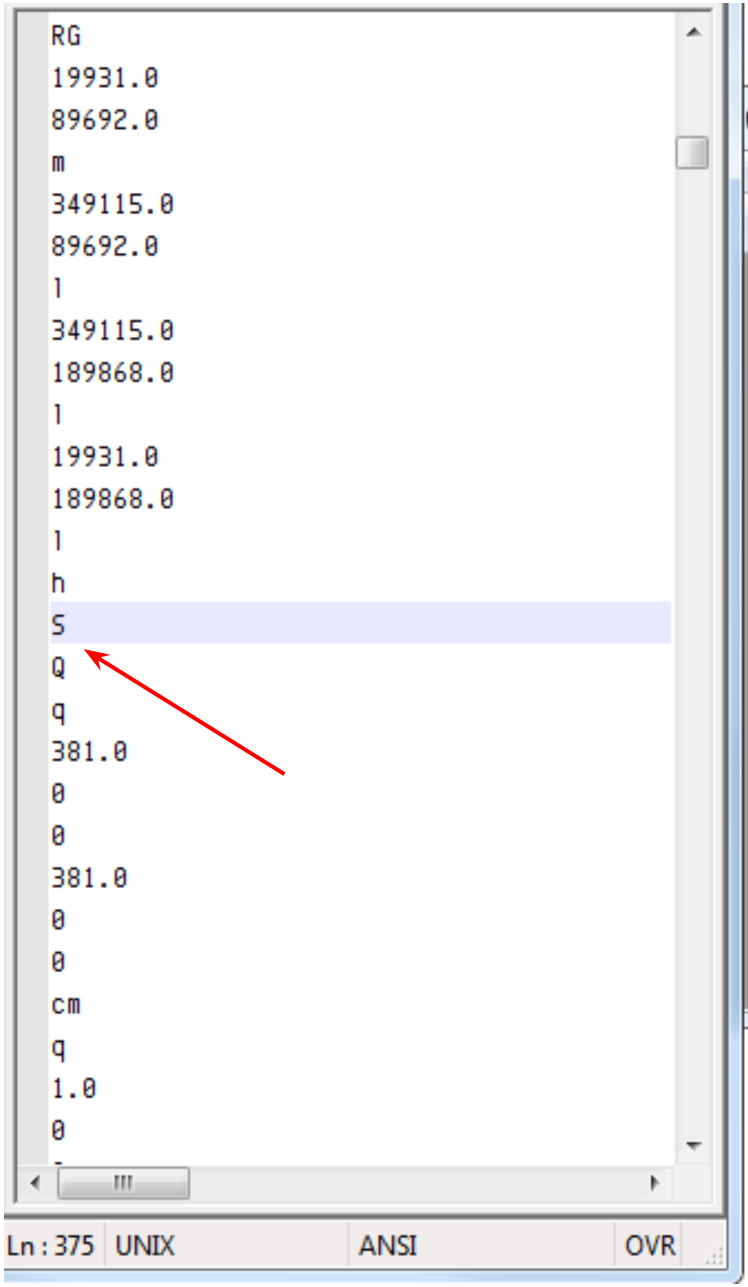


Operands	Operator	Description
string	**Tj**	Show a text string.
string	**'**	Move to the next line and show a text string. This operator shall have the same effect as the code T* *string* Tj
a_w a_c *string*	**"**	Move to the next line and show a text string, using a_w as the word spacing and a_c as 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**.

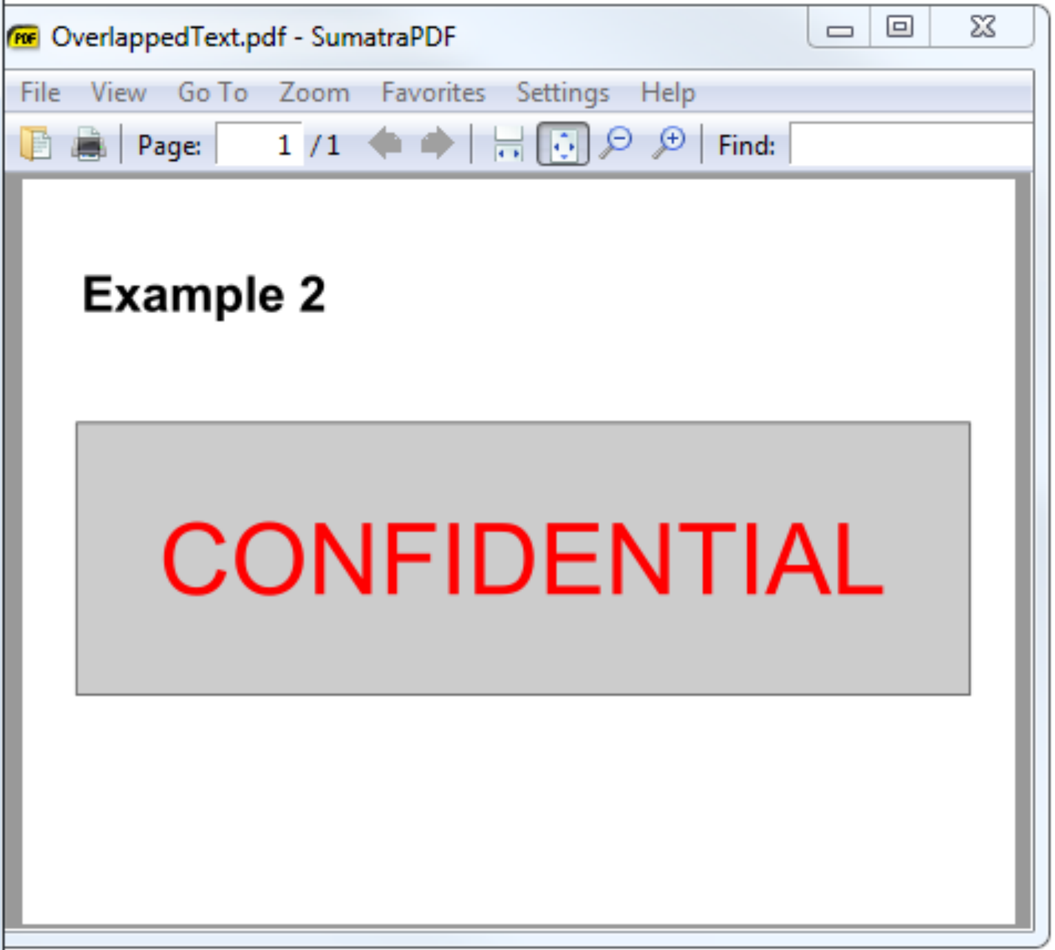
text showing operators

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

Example:
manually remove
overlapping elements



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



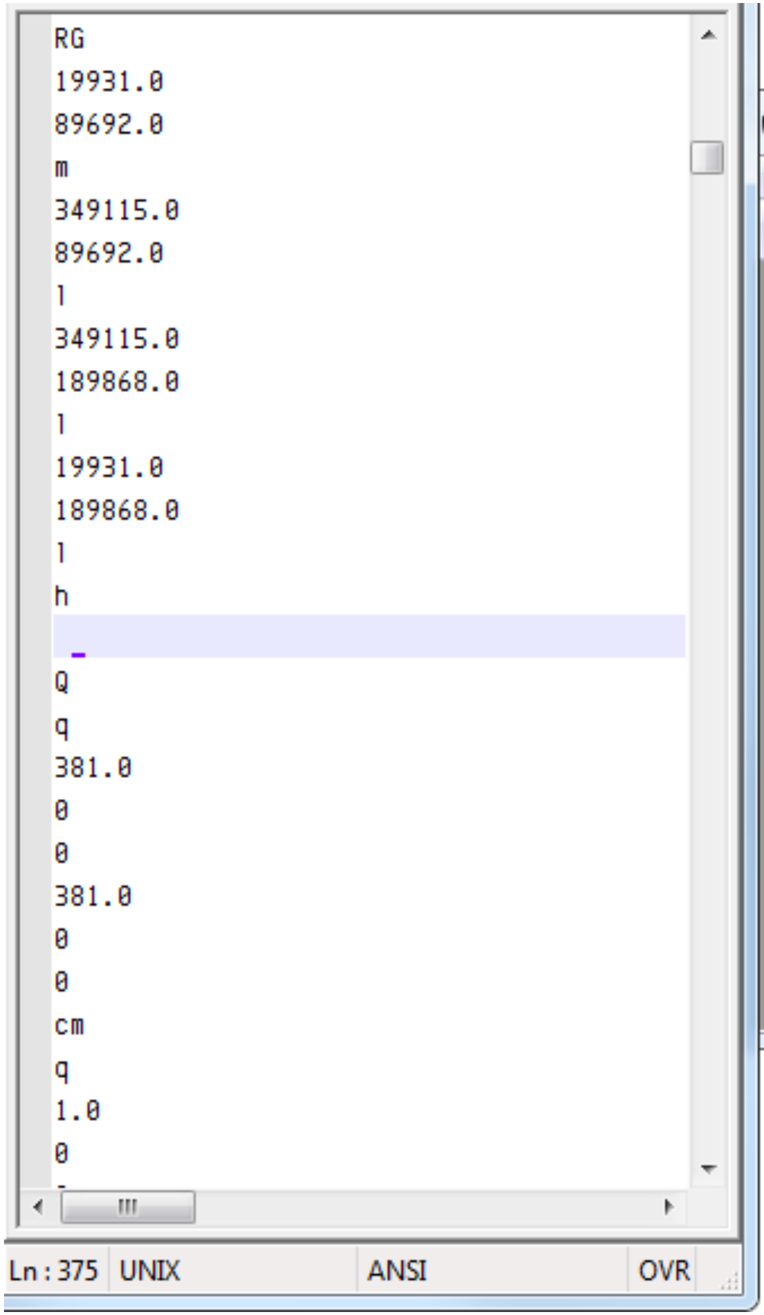
Example 2

CONFIDENTIAL

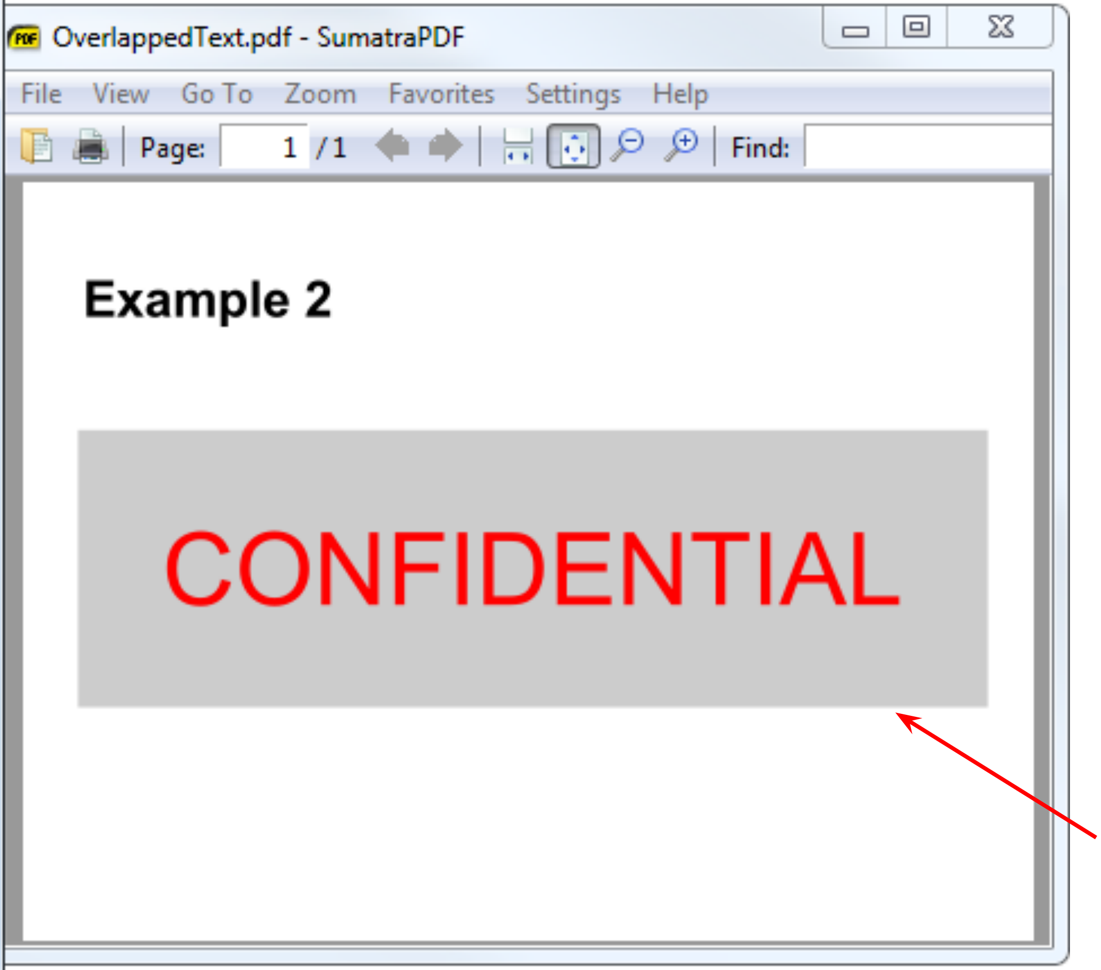
take the uncompressed PDF

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
```

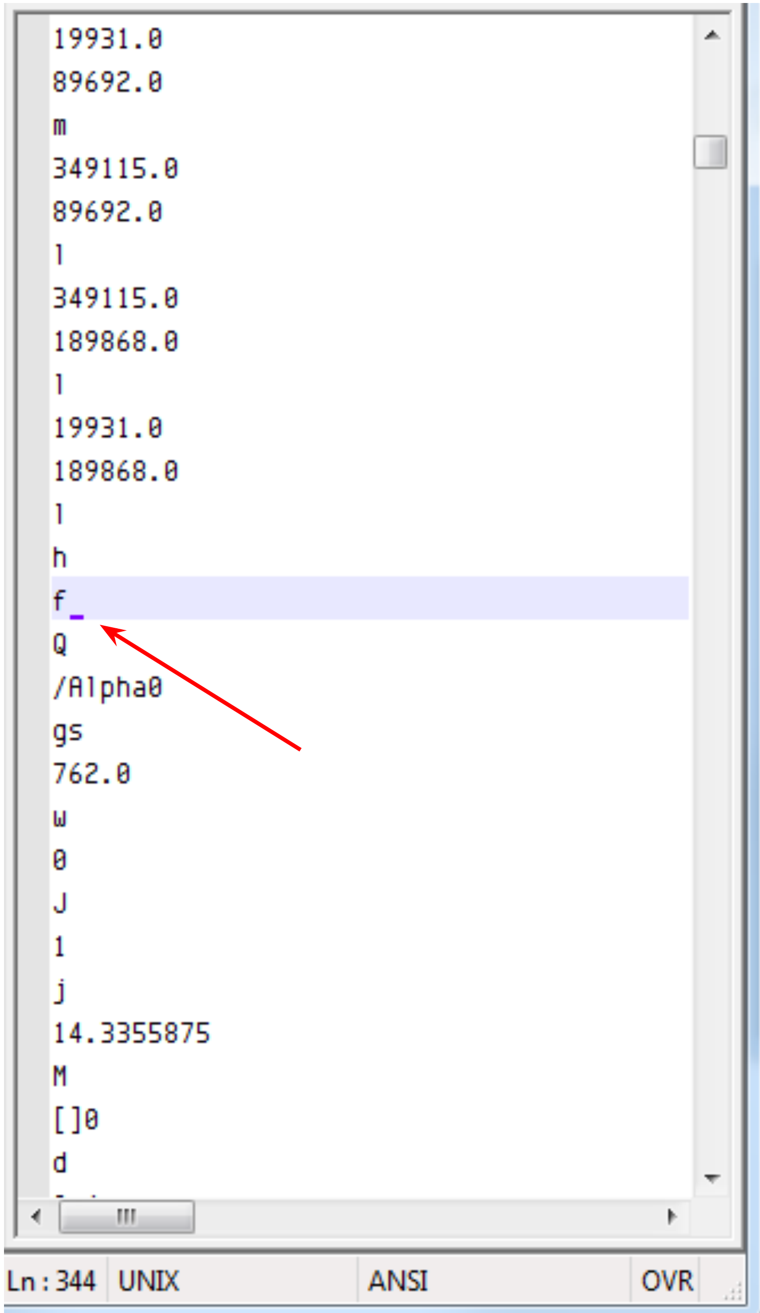


Example 2

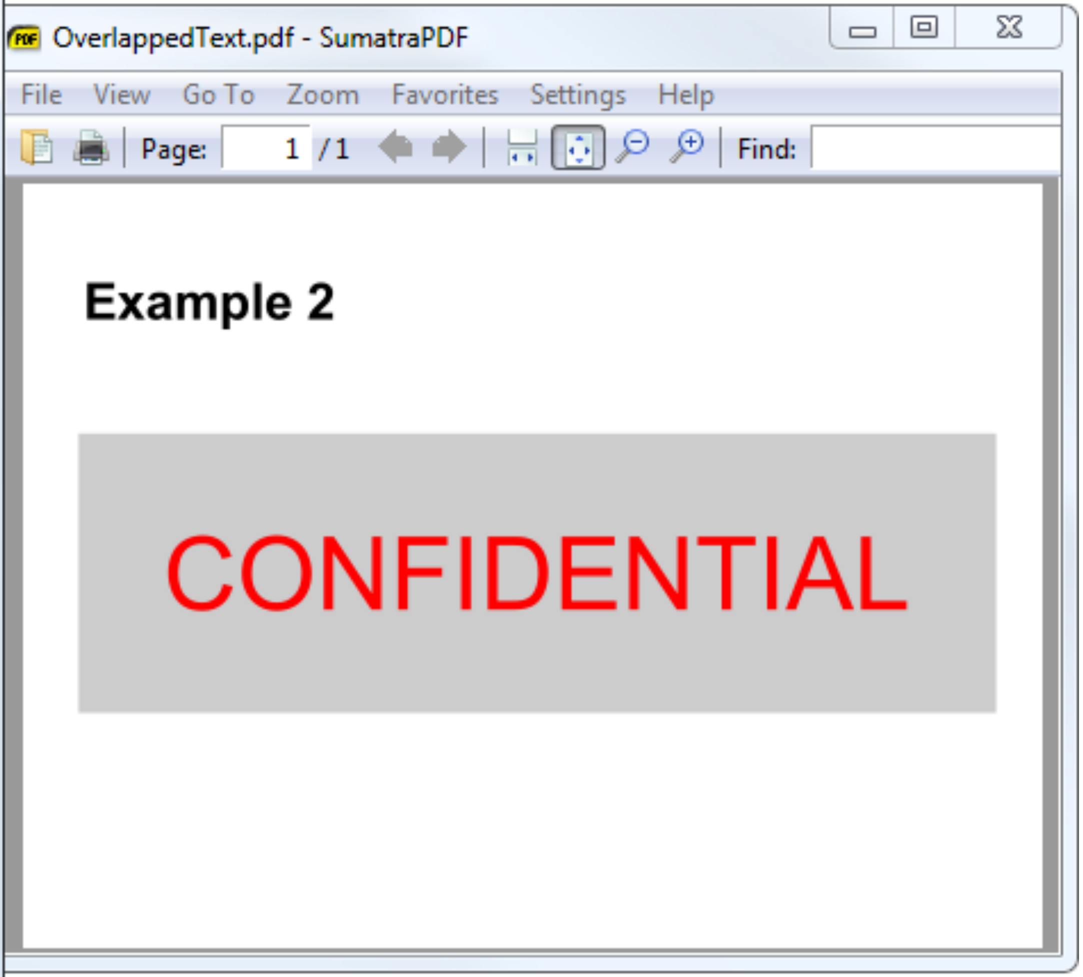
CONFIDENTIAL

erase the S

⇒ no more black border



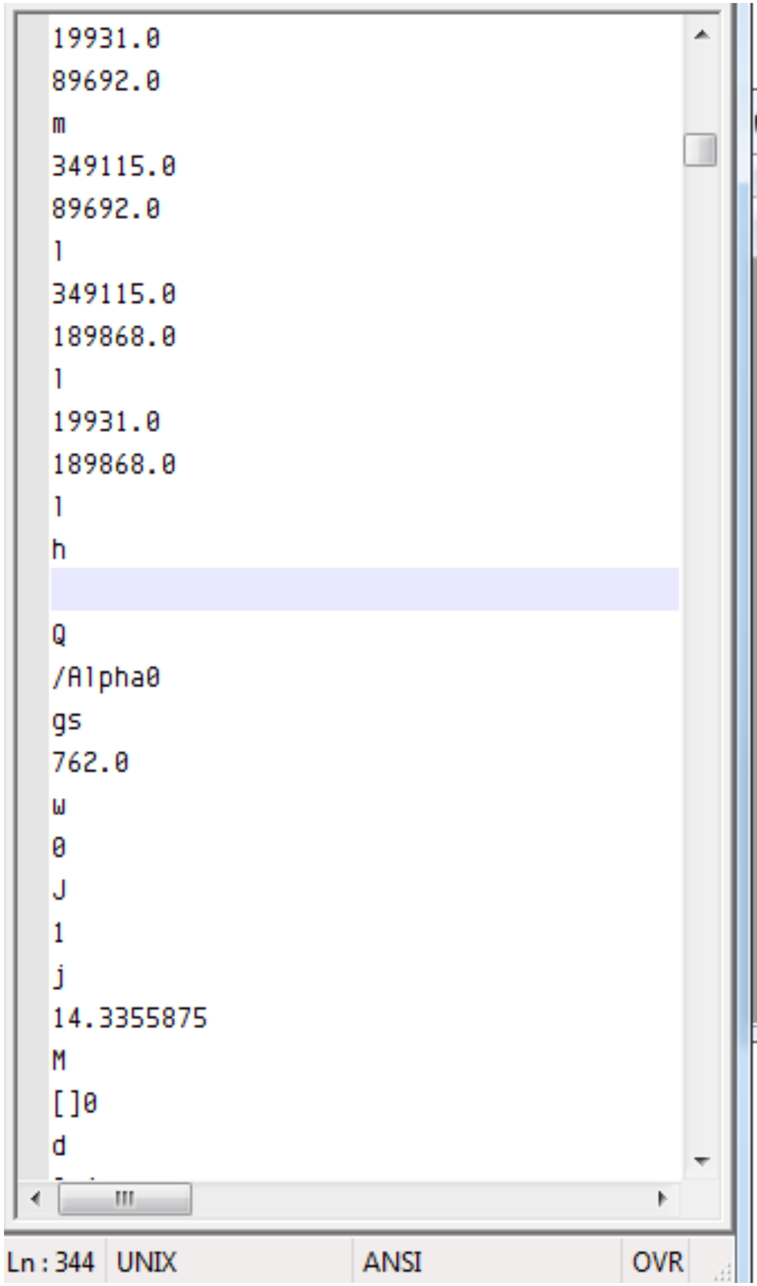
```
19931.0
89692.0
m
349115.0
89692.0
l
349115.0
189868.0
l
19931.0
189868.0
l
h
f
Q
/Alpha0
gs
762.0
w
0
J
1
j
14.3355875
M
[]0
d
```



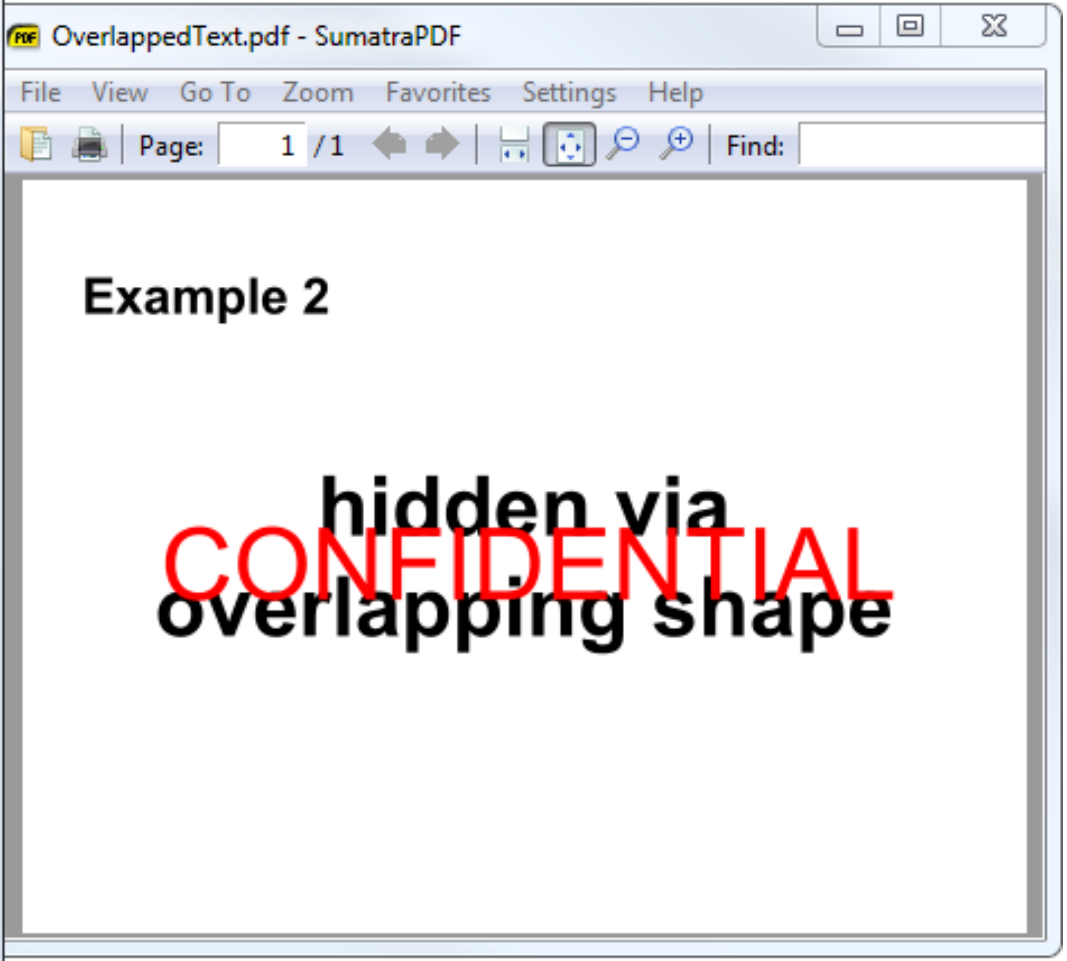
Example 2

CONFIDENTIAL

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



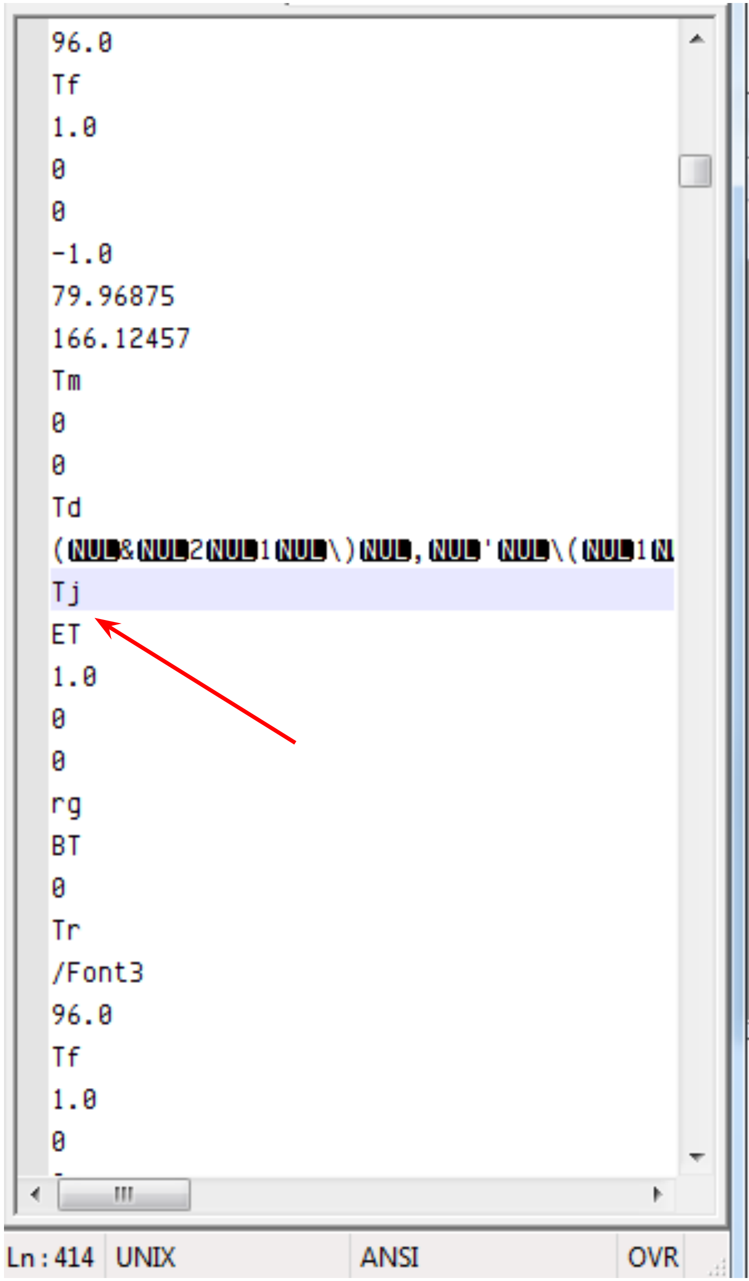
```
19931.0
89692.0
m
349115.0
89692.0
l
349115.0
189868.0
l
19931.0
189868.0
l
h

Q
/Alpha0
gs
762.0
w
0
J
1
j
14.3355875
M
[]0
d
```



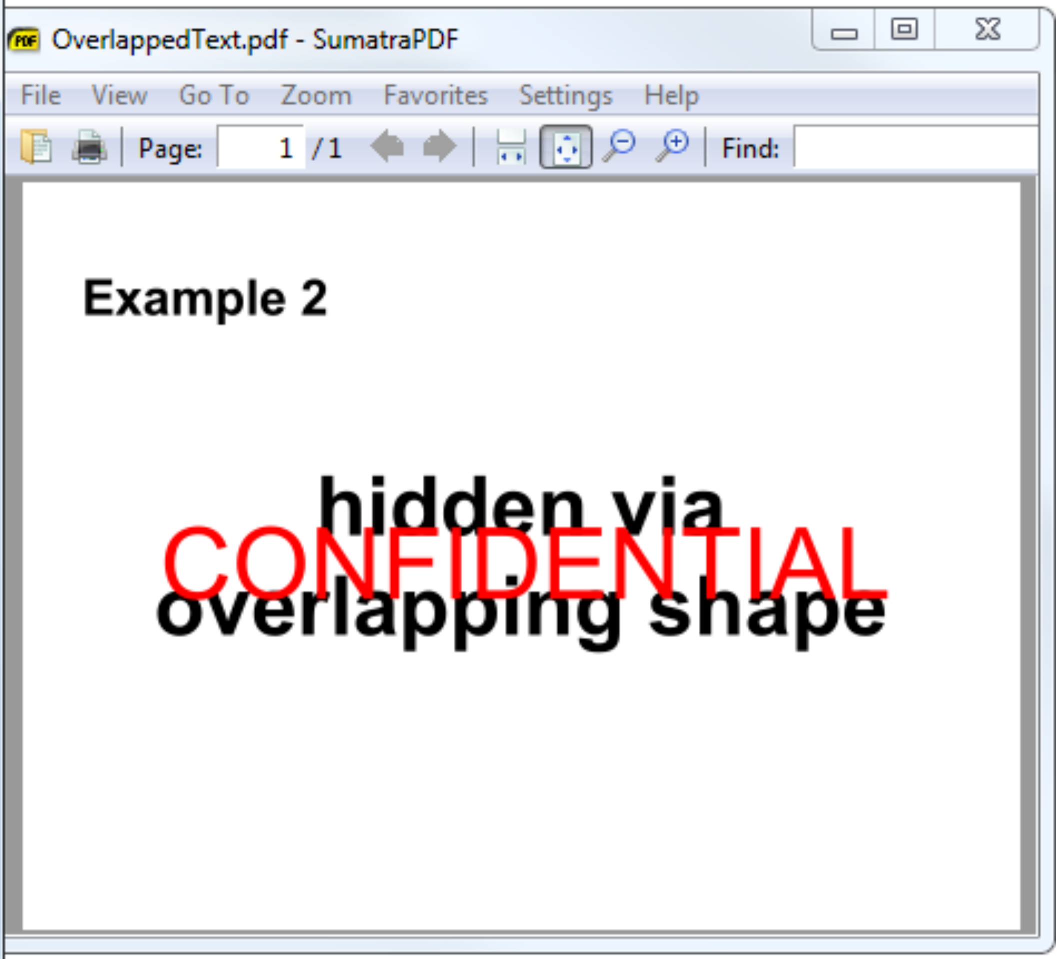
Example 2

hidden via
CONFIDENTIAL
overlapping shape

⇒ no more gray surface



```
96.0
Tf
1.0
0
0
-1.0
79.96875
166.12457
Tm
0
0
Td
(NUL&NUL2NUL1NUL\)NUL,NUL'NUL\(NUL1N
Tj
ET
1.0
0
0
rg
BT
0
Tr
/Font3
96.0
Tf
1.0
0
```



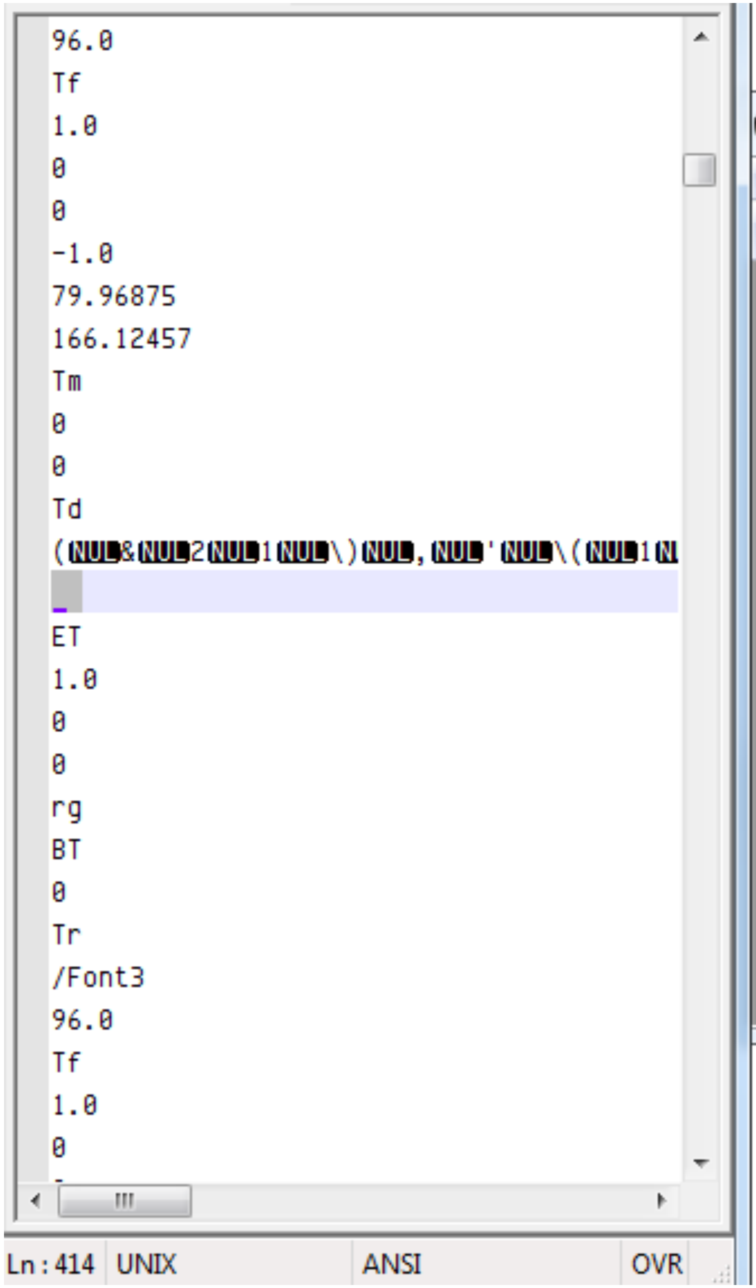
Example 2

hidden via
CONFIDENTIAL
overlapping shape

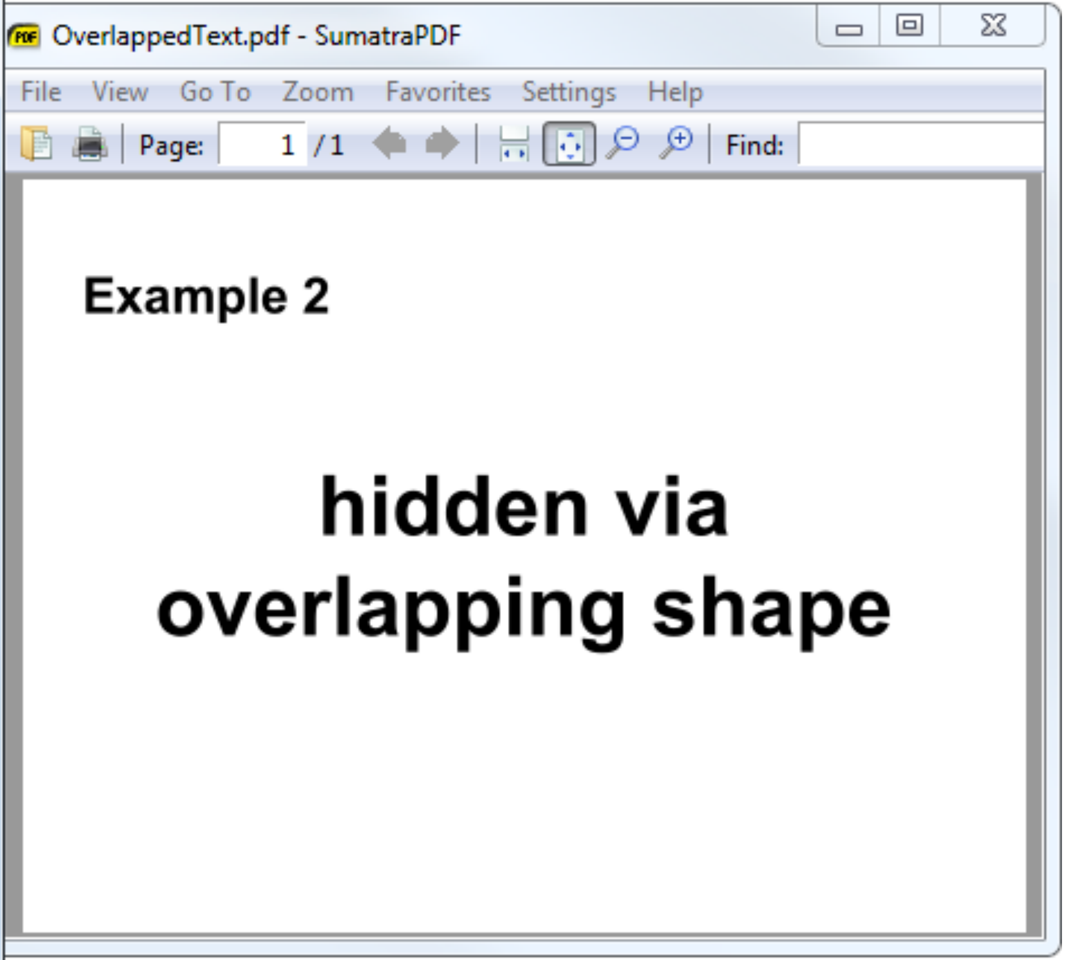
and the “obvious” `Tj` after the string `(...)`

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

&→C, 2→O, 1→N...



```
96.0
Tf
1.0
0
0
-1.0
79.96875
166.12457
Tm
0
0
Td
(NUL&NUL2NUL1NUL\)NUL,NUL'NUL\(NUL1N

ET
1.0
0
0
rg
BT
0
Tr
/Font3
96.0
Tf
1.0
0
```



Example 2

hidden via overlapping shape

→ no more hidden elements!

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

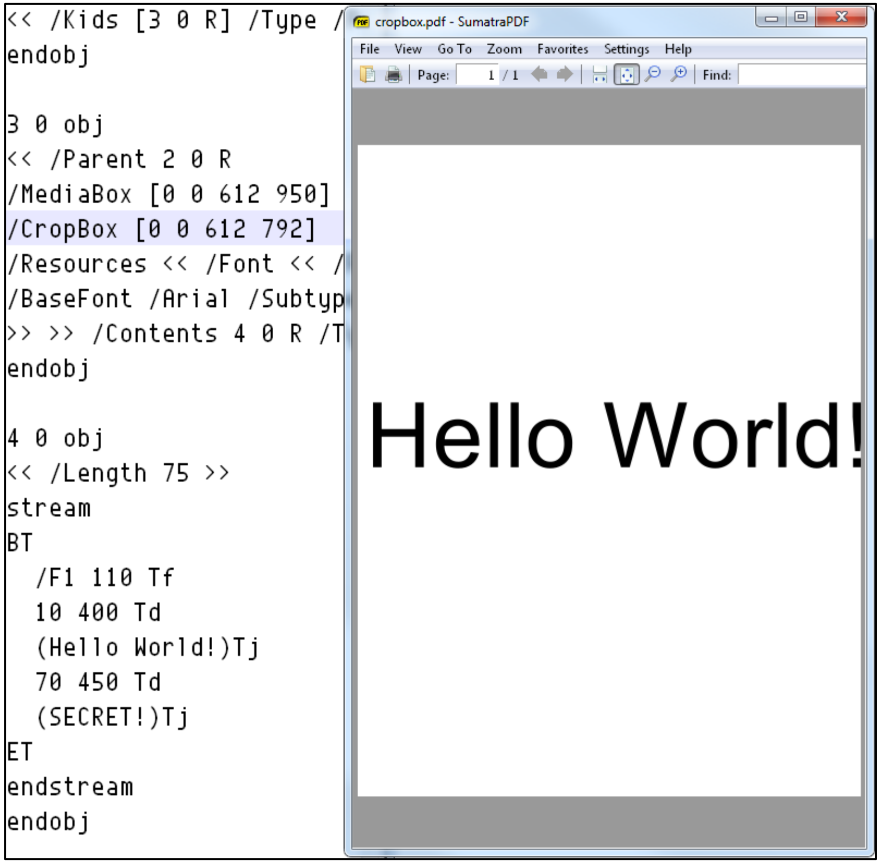
Page size tricks

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

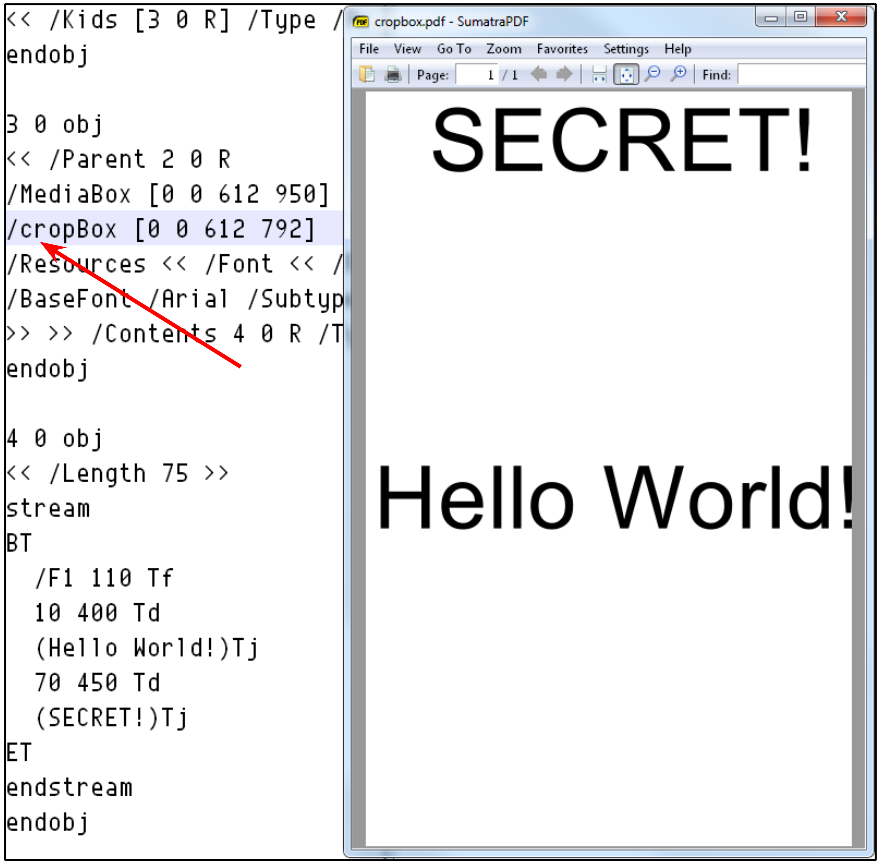
- What you see is `/CropBox`
  - Copy/Paste and (some) `pdftotext` respect that

⇒ what is in `Mediabox` (but not `CropBox`)

is not extracted

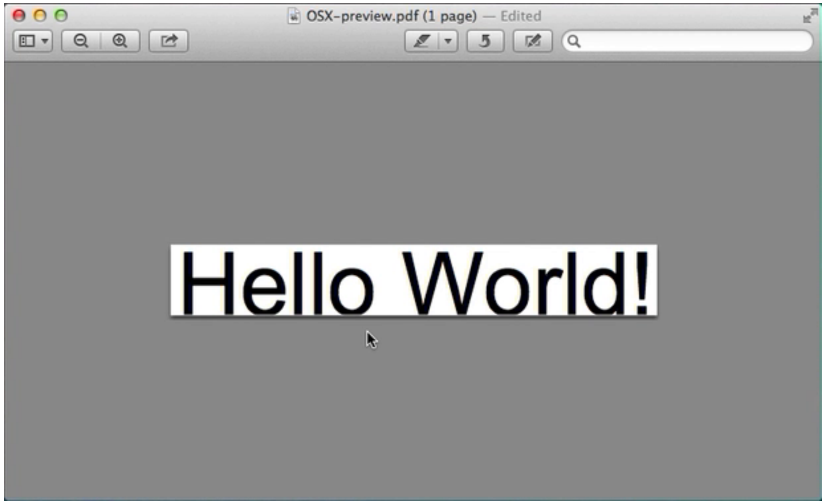


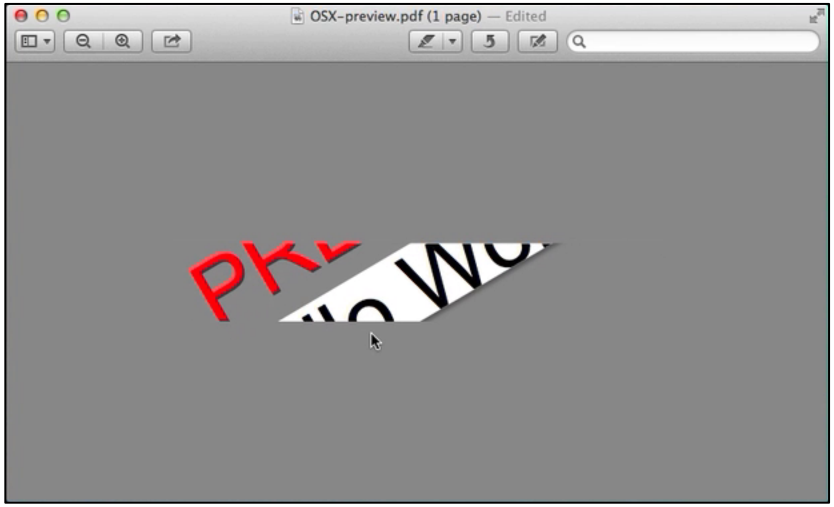
```
<< /Kids [3 0 R] /Type /
endobj

3 0 obj
<< /Parent 2 0 R
/MediaBox [0 0 612 950]
/CropBox [0 0 612 792]
/Resources << /Font << /
/BaseFont /Arial /Subtyp
>> >> /Contents 4 0 R /T
endobj

4 0 obj
<< /Length 75 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!)Tj
  70 450 Td
  (SECRET!)Tj
ET
endstream
endobj
```

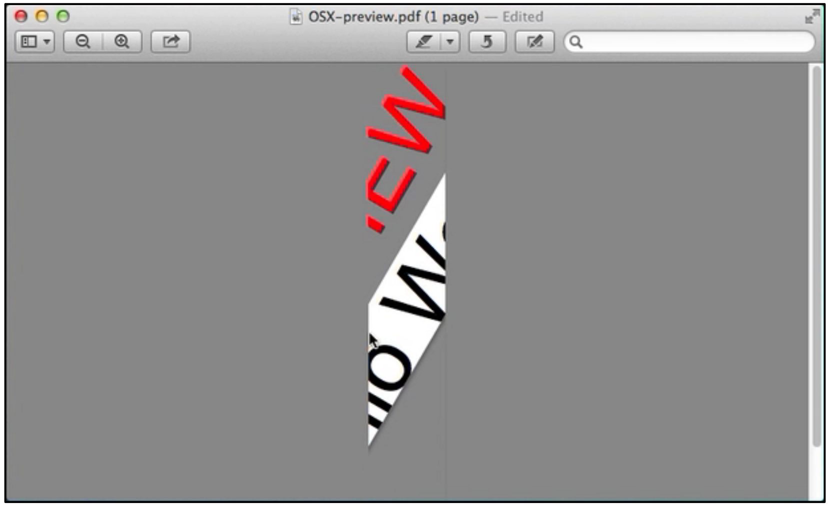


```
<< /Kids [3 0 R] /Type /
endobj

3 0 obj
<< /Parent 2 0 R
/MediaBox [0 0 612 950]
/cropBox [0 0 612 792]
/Resources << /Font << /
/BaseFont /Arial /Subtyp
>> >> /Contents 4 0 R /T
endobj

4 0 obj
<< /Length 75 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!)Tj
  70 450 Td
  (SECRET!)Tj
ET
endstream
endobj
```

disable /CropBox to see the full contents



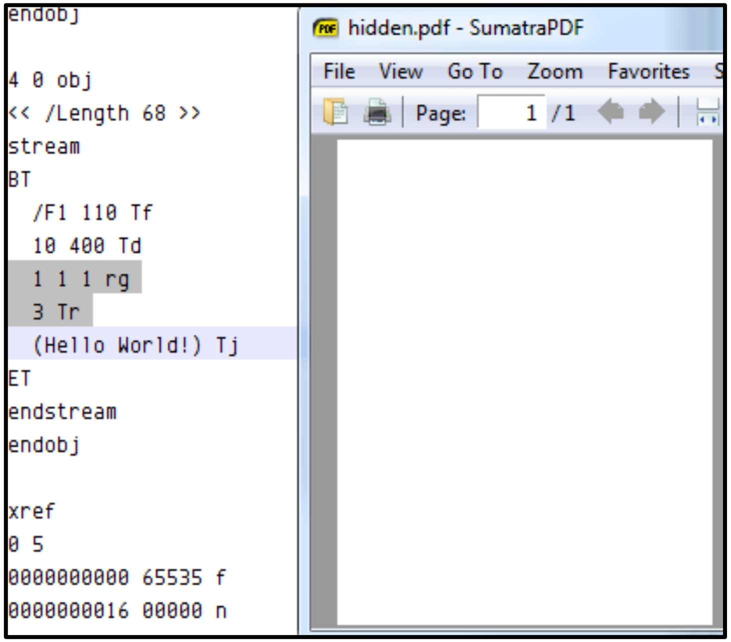




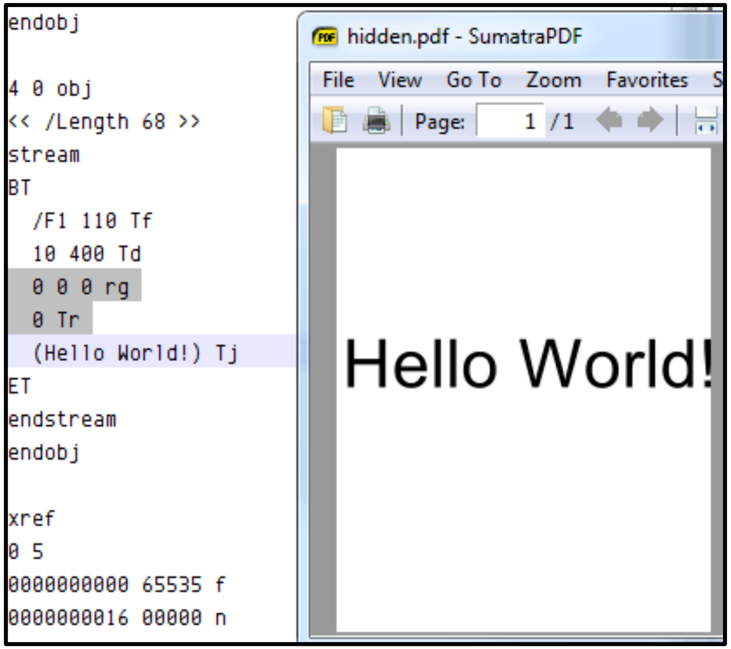
OS-X actually does a /CropBox when you copy/paste out of a PDF, and you can see the full original content by rotating the page.

Hidden text

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



```
endobj

4 0 obj
<< /Length 68 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  1 1 1 rg
  3 Tr
  (Hello World!) Tj
ET
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
```



```
endobj

4 0 obj
<< /Length 68 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  0 0 0 rg
  0 Tr
  (Hello World!) Tj
ET
endstream
endobj

xref
0 5
0000000000 65535 f
0000000016 00000 n
```

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

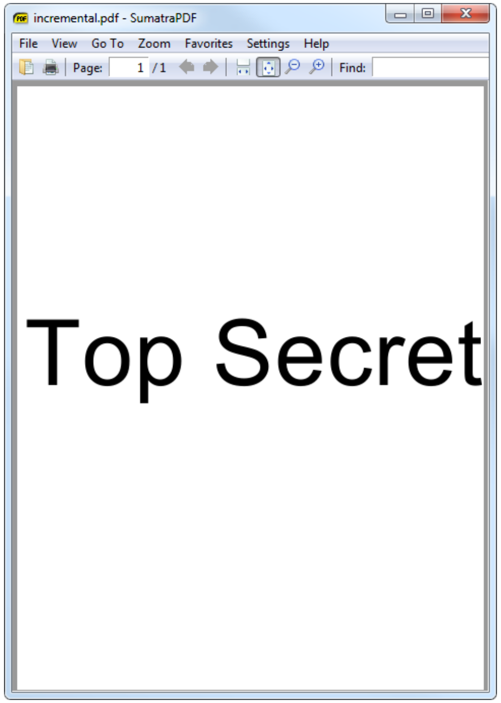
the concept:

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


incremental.pdf - SumatraPDF
File View Go To Zoom Favorites Settings Help
Page: 1 /1
Find:

Top Secret

```
%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

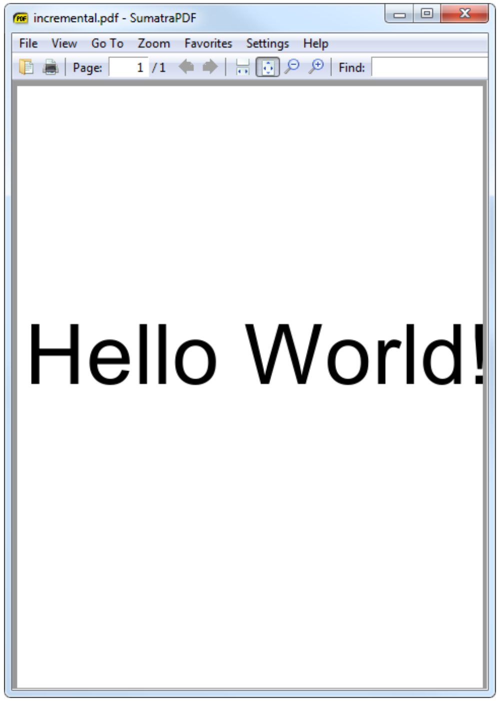
points to the new **xref**

```
startxref
654
%%EOF
```

Result

⇒ different content !

restore content by cutting after the first %%EOF



incremental.pdf - SumatraPDF
File View Go To Zoom Favorites Settings Help
Page: 1 / 1
Find:

Hello World!

Incremental update to hide page

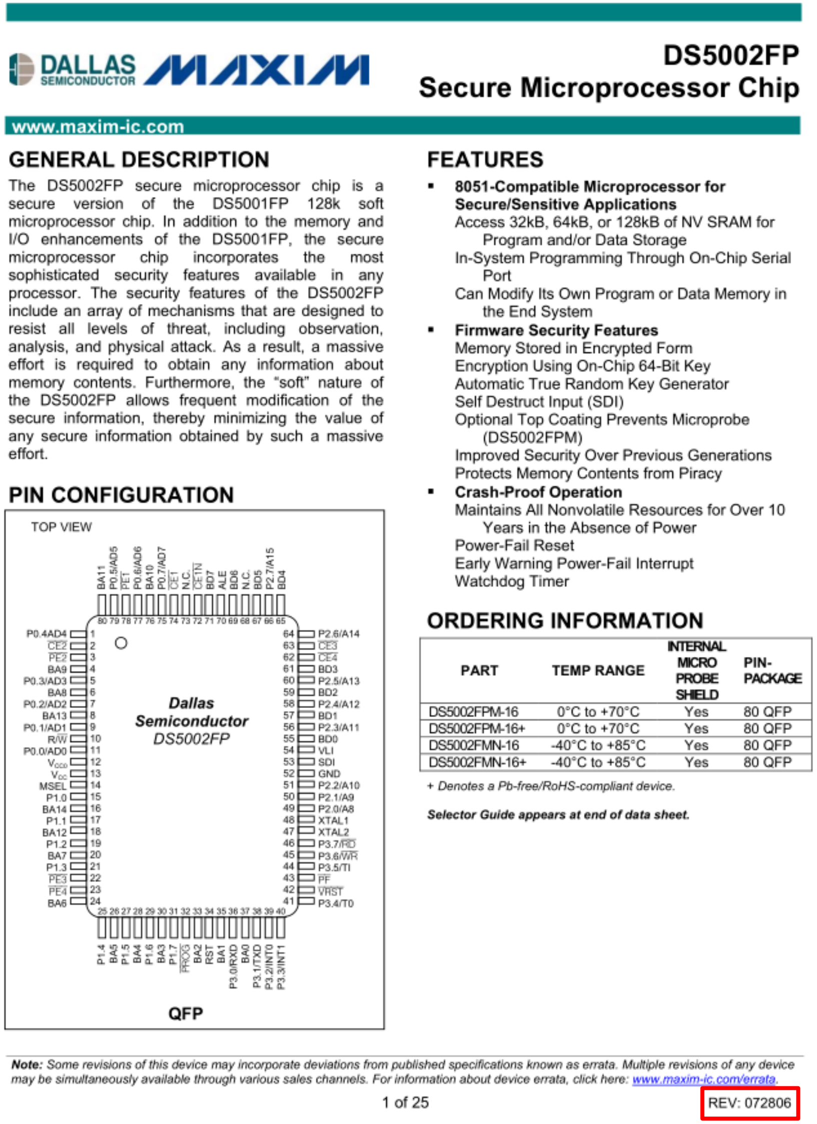
use the same trick

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

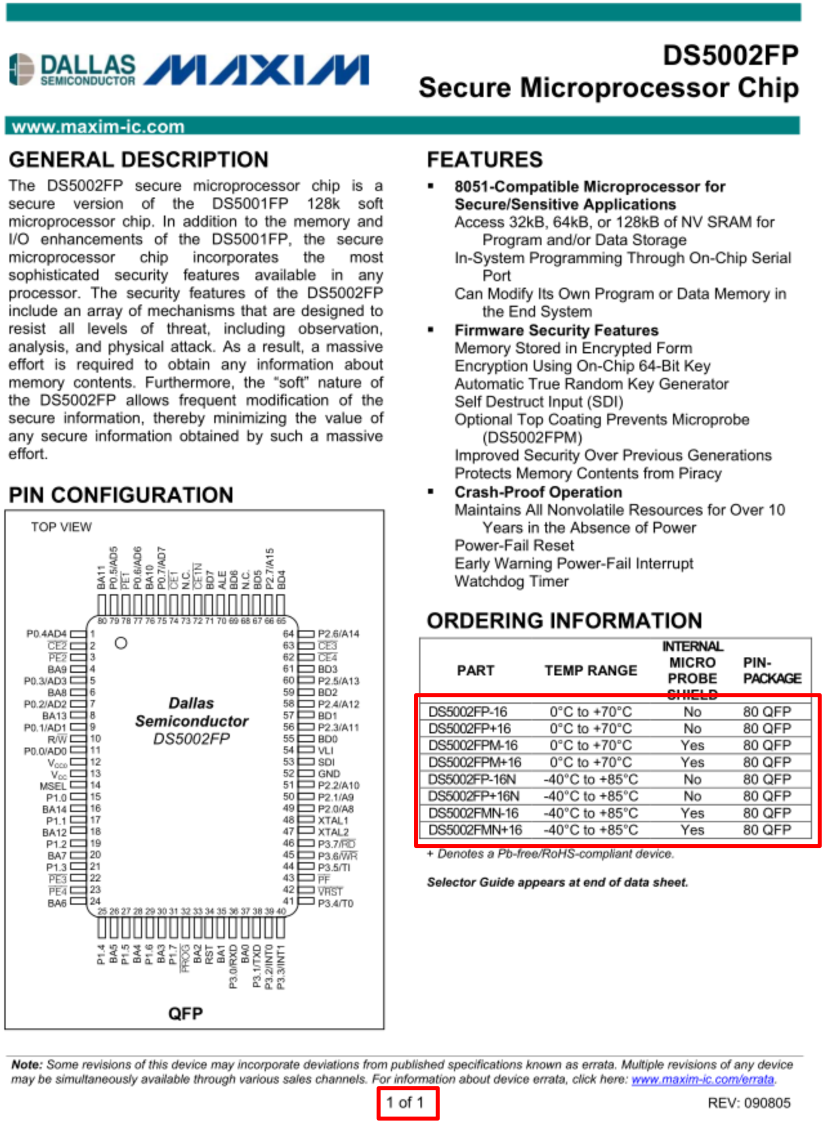
PART	TEMP RANGE	INTERNAL MICRO PROBE SHIELD	PIN-PACKAGE
DS5002FPM-16	0°C to +70°C	Yes	80 QFP
DS5002FPM-16+	0°C to +70°C	Yes	80 QFP
DS5002FMN-16	-40°C to +85°C	Yes	80 QFP
DS5002FMN-16+	-40°C to +85°C	Yes	80 QFP

+ 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.*

1 of 25 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.

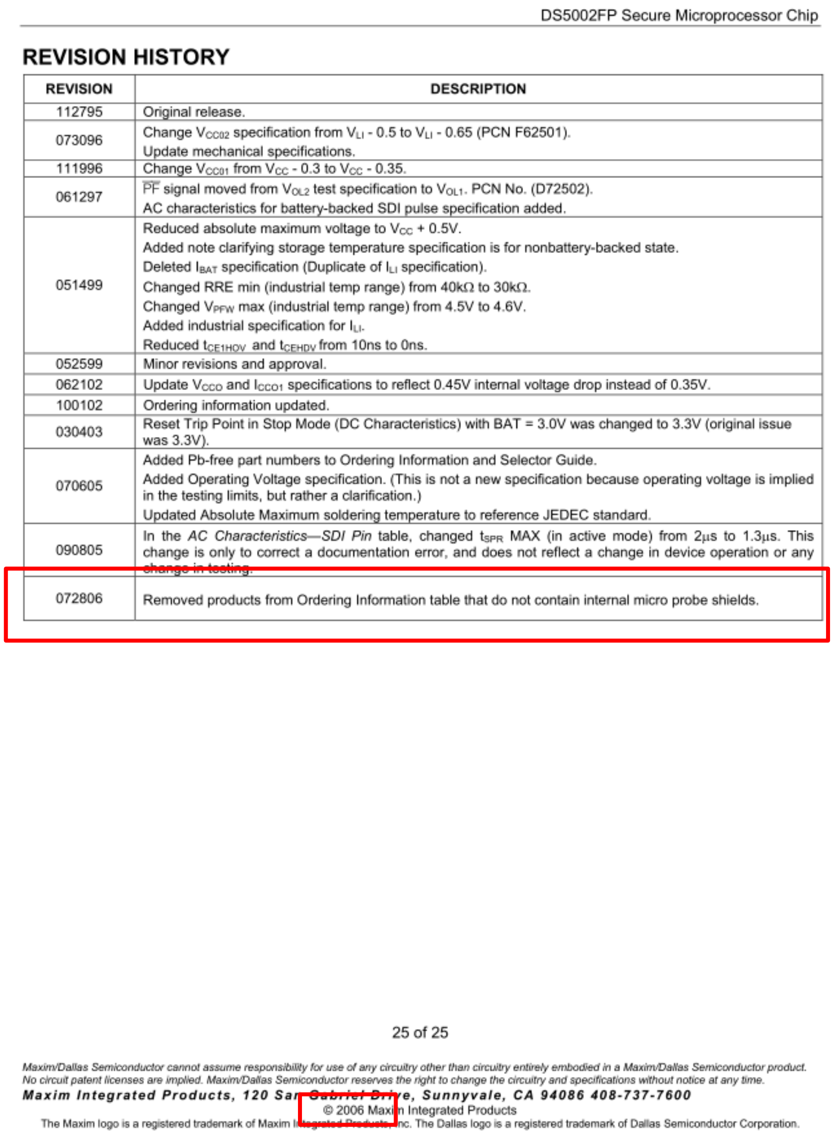
Selector Guide appears at end of data sheet.

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.

1 of 1 REV: 090805

incremental PDF found in the wild

(removed parts, incorrect page number)



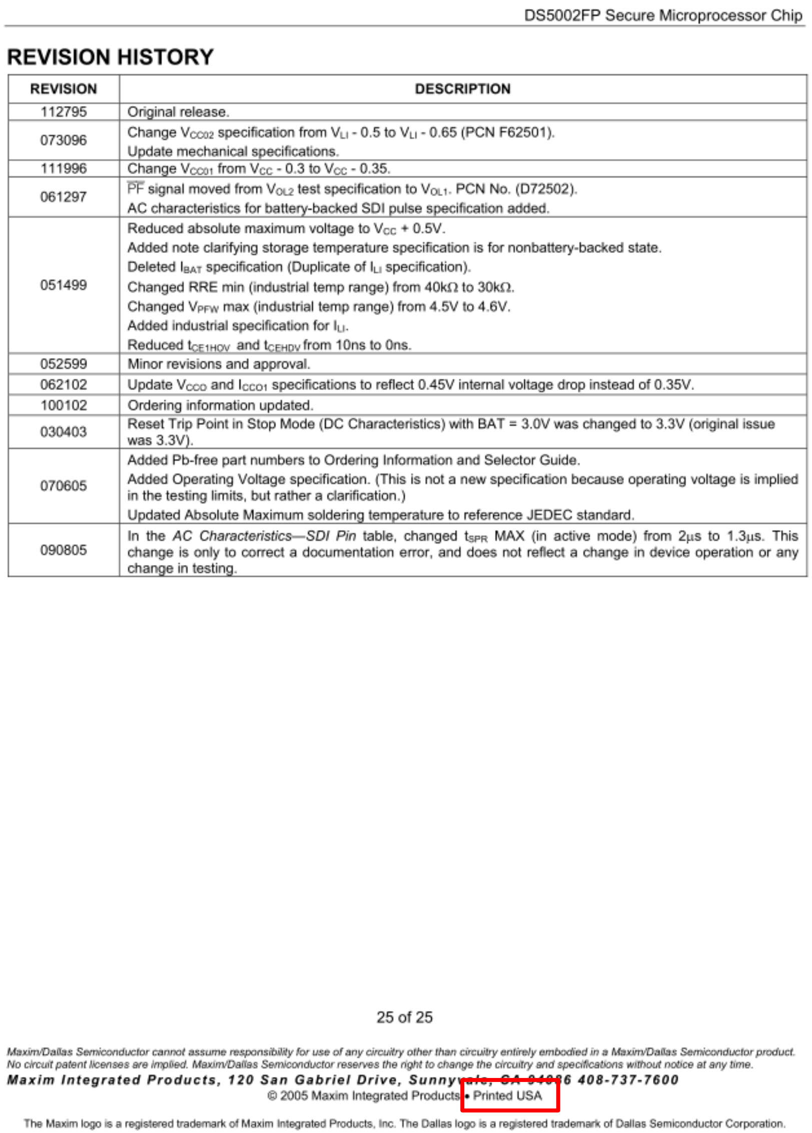
REVISION HISTORY

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/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

The Maxim logo is a registered trademark of Maxim Integrated Products, Inc. The Dallas logo is a registered trademark of Dallas Semiconductor Corporation.



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.

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

The Maxim logo is a registered trademark of Maxim Integrated Products, Inc. The Dallas logo is a registered trademark of Dallas Semiconductor Corporation.

“Printed USA”

Copy/Paste corruption

some files produced corrupted text when copying

(mentioned in the first part)

this is due to fonts:

- ○ `/Subtype /Type3`
- ○ with no `/ToUnicode` mapping

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

@angealbertini

corkami.com