pdfSweep

an iText 7 add-on

itextpdf.com

pdfSweep: what it does and why you need it

Today, it is easy to distribute documents among large numbers of people with diverse needs. You can also grant or restrict access the content of these documents to keep information in the right hands. Both company policies and international regulations may require redaction of sensitive information before archiving or sharing documents with all users. Companies and organizations need tools to implement this redaction process in a flexible and automated way, with absolute guarantees regarding the elimination of the targeted elements. pdfSweep can help.

What is pdfSweep?

pdfSweep is an iText add-on that removes (redacts) sensitive information from a PDF (Portable Document Format) document. Confidentiality is assured, because the redacted information cannot be recovered. In a secure two-step process, pdfSweep deletes text and images at user-defined coordinates, or as defined by a regular expression. After having parsed the rendering information in the original PDF document, a new document is created without the redacted content.

Why do we need redaction?

Redaction can be useful whenever the publisher or author of a document wishes to take out certain information. Common use cases include:

- Freedom of Information Act (USA) and similar legislation in other countries
- Governmental declassification procedures
- Proprietary information
- Trade secrets
- General Data Protection Regulation (Europe)
- All data that would impact the privacy of people
  - Social security numbers (USA)
  - National register identification numbers
  - Phone numbers
  - Bank account details
  - Names of people in a clinical trial

A short history of redaction

In the past, 'redaction' simply involved printing a document, blacking out the necessary information and making a photocopy of the document. That way, all information covered by dark ink simply does not get copied. This worked because paper is a simple WYSIWYG format. There is no hidden data, no metadata that needs to be erased.

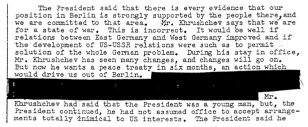
The President said that there is every evidence that our position in Berlin is strongly supported by the people there, and we are committed to that area. Mr. Khrushchev says that we are for a state of war. This is incorrect. It would be well if relations between East Germany and West Germany improved and if the development of US-USSR relations were such as to permit solution of the whole German problem. During his stay in office, Mr. Khrushchev has seen many changes, and changes will go on. But now he wants a peace treaty in six months, an action which would drive us out of Berlin. [redacted] Mr. Khrushchev had said that the President was a young man, but, the President continued, he had not assumed office to accept arrangements totally inimical to US interests. The President said he

Figure 0: Example redacted document

In addition to the content as such, a PDF document essentially contains instructions for rendering the document in a viewer. Adding an instruction to draw a black rectangle does not erase text-rendering instructions pertaining to the information underneath the rectangle. This essentially means that covering up information is no longer sufficient, and actual removal/replacement of the content and related instructions is needed. Otherwise text-extraction (or simply copy/pasting the text from your viewer to a text editor) would yield the 'redacted' but still available data.

Challenges in redacting PDF documents

This is a (well-behaved) example of raw PDF content. It tells the viewer to render the text "appearance" and "Pilot".

```
[a, -28.7356, p, 27.2652, p, 27.2652, e, -27.2652, a, -28.7356, r, 64.6889, a, -28.7356,
n, 27.2652, c, -38.7594, e, 444] TJ
/R10 10.44 Tf
68.16 0.24 Td
[P, -18.7118, i, -9.35592, l, -9.35592, o, -17.2414, t, -9.35636] TJ
```

In general, redacting a PDF document is significantly harder than simply covering up the data that is to be redacted because:

- Text rendering instructions do not need to appear in logical (reading) order.
- Text rendering instructions do not always constitute complete words/chunks of text.
- Transformations can be applied (shift, scale, font transformation, etc).
- PDF documents can contain metadata, which should also be checked.
- PDF documents can contain scripts. Thus adding the possibility of dynamic content.

Similar problems occur with images:

- Images can be defined as a series of drawing operations.
- All instructions of a page have to be processed to determine where each shape intersects with the redaction area, in order to find a neat way to avoid drawing in that area.
- Images can be added to a PDF document under many formats.

How does pdfSweep work?

The basic pdfSweep workflow has just two easy steps:

- Select those parts of the document that must be redacted: either by specifying the coordinates, or by inputting a regular expression that fits your needs. We have already provided a substantial list of common regular expressions to do some of the heavy lifting for you, such as social security numbers, phone numbers, dates, etc.
- Pass the locations to pdfSweep, or invoke pdfAutoSweep with the pattern(s) of your choice.

This is a pdfAutoSweep example that redacts the words 'Alice' and 'White Rabbit' and 'Rabbit' (regardless of casing). It marks all occurrences of Alice with a pink rectangle, and all occurences of 'Rabbit' with a gray rectangle.

```
// Load the license to be able to use pdfSweep
LicenseKey.loadLicenseFile(licenceFile);

String input = "AliceInWonderLand.pdf";
String output = "AliceInWonderLandRedacted.pdf";

// define a composite strategy
CompositeCleanupStrategy strategy = new CompositeCleanupStrategy();
strategy.add(new RegexBasedCleanupStrategy("Alice").setRedactionColor(Color.PINK));
strategy.add(new RegexBasedCleanupStrategy("(White )*Rabbit").setRedactionColor(Color.
GRAY));

PdfWriter writer = new PdfWriter(output);
writer.setCompressionLevel(0);
PdfDocument pdf = new PdfDocument(new PdfReader(input), writer);

// sweep
PdfAutoSweep pdfAutoSweep = new PdfAutoSweep(strategy);
pdfAutoSweep.cleanUp(pdf);
```


This is the original document:

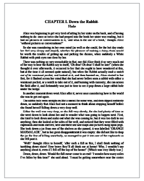

Figure 1: pdfSweep original input document

And this is 'post redaction':

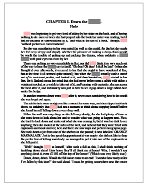

Figure 2: pdfSweep redacted output document

As is made clear by this example document (and the code to go with it), it is perfectly possible to define a custom color for each snippet of text to be redacted.

Similarly, you could also have specified the exact locations yourself. Following code-snippet demonstrates that use case:

```
String input = "iphone_user_guide_untagged.pdf";
String output = "iphone_user_guide_redacted.pdf";

// load licensekey
LicenseKey.loadLicenseFile("itext_dev_master_license.xml");

// define rectangles
List<Rectangle> rects = Arrays.asList( new Rectangle(60f, 80f, 460f, 65f),
                              new Rectangle(300f, 370f, 215f, 260f)
                            );

// turn rectangles into cleanup locations
int pagesNum = 130;
List<PdfCleanUpLocation> cleanUpLocations = new ArrayList<>();
for (int i = 0; i < pagesNum; ++i) {
  for (int j = 0; j < rects.size(); ++j) {
     cleanUpLocations.add(new PdfCleanUpLocation(i + 1, rects.get(j)));
  }
}

// open document
PdfDocument pdfDocument = new PdfDocument(new PdfReader(input), new PdfWriter(output));

// perform cleanup
PdfCleanUpTool cleaner = new PdfCleanUpTool(pdfDocument, cleanUpLocations);
cleaner.cleanUp();

// close document
pdfDocument.close();
```

Alternative example

In an alternative workflow, you can add human verification:

- Select those parts of the document that must be redacted.
- Call pdfAutoSweep with the method tentativeCleanUp. This will produce a document where the content was not actually redacted, but where all content that supposedly must be redacted is marked with an annotation. Adobe Acrobat can manage these annotations for you, allowing you to remove redaction annotations, add extra annotations, and add comments.
- Once you have made your final selection about which redaction annotations to keep, you can hit "apply redaction" in Adobe Acrobat.

'Redacting' Image elements

The previous examples mostly showcase how to use pdfSweep on text. Redaction can also be performed on images. Images are automatically redacted when they intersect with the rectangles the user provided. The redacted areas are covered with a colored rectangle, and the underlying image is changed (to prevent image extraction from yielding the original image.)

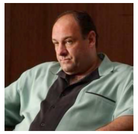

Figure 3: Tony Soprano as displayed in the original PDF document

Figure 4: Tony Soprano as displayed after the eyes have been redacted to provide anonymity

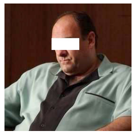

Figure 5: Extracted image of Tony Soprano. Notice that the original content is gone

How does pdfAutoSweep work?

1. The end user can specify a regular expression, and optionally a color.
2. The document is processed a first time, all instructions in the PDF document that relate to text rendering are processed. All characters along with their bounding rectangles in the document are stored.
3. These intermediate data structures are sorted so that all characters are now in logical (reading) order.
4. The regular expression(s) provided by the user are matched.
5. The information about where the match took place, and the bounding rectangles of the characters involved provide pdfSweep with the rectangles that need to be redacted.

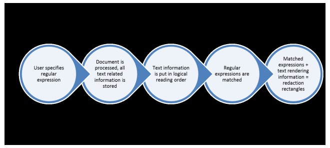

Figure 6: pdfAutoSweep workflow

Pre-configured regular expressions

Most of the regular expressions that are already configured to work with pdfAutoSweep are inspired by the O' Reilly reference work "Regular Expressions Cookbook" (Jan Goyvaerts and Steven Levithan, 978-1-449-31943-4). What follows is a short listing of some of these regular expressions and their intended matches.

NAME	MATCHES
ROMAN_NUMERALS_STRICT	a Roman number, matching only those numbers that are valid.
ROMAN_NUMERALS_FLEXIBLE	a Roman number, also matching IIII (4), whereas STRICT only matches IV
US_SOCIAL_SECURITY_NUMBER	a US social security number (ddd-dd-dddd)

NAME	MATCHES
US_ZIP_CODE	a US zip code (either 5 digit format, or 5+4 format)
US_CURRENCY	an amount of US dollars
CANADA_ZIP_CODE	a Canadian zip code
UK_ZIP_CODE	a UK zip code
DATE_MM_DD_YYYY	a date, specified in MM/DD/YYYY, separator can be 'space' '/' or '-'
DATE_MM_DD_YYYY_HH_MM_SS	a date, specified in MM/DD/YYYY HH:MM:SS
DATE_DD_MM_YYYY	a date, specified in DD/MM/YYYY, separator can be 'space' '/' or '-'
DATE_DD_MM_YYYY_HH_MM_SS	a date, specified in MM/DD/YYYY HH:MM:SS
IPV4_ADDRESS	an Internet Protocol (v4) address
IPV6_ADDRESS	an Internet Protocol (v6) address
MAC_ADDRESS	a Media Access Control address
EMAIL_ADDRESS	an email address
HTTP_URL	a url

Performance

To give a brief idea of the performance of pdfSweep, we've used the iPhone user manual as a reference. We've redacted the regular expression "(i|I)Phone". You can imagine this occurs quite a lot in that document.

Where "copies" signifies how many times the original input document was concatenated with itself. E.g. 4 copies means the document (of 130 pages) was concatenated 4 times (resulting in 520 pages).

Here are the results

COPIES	#PAGES	#INPUT FILE SIZE (MB)	#MILLISECONDS
1	130	3.23	6407
2	260	5.55	10253
4	520	11.1	17946
8	1040	22.5	23445

From this table we can clearly see pdfSweep performs linearly in relation to the size of the input document.

Conclusion

In this whitepaper, we've briefly presented our add-on pdfSweep. pdfSweep allows you to seamlessly integrate data redaction in your existing workflow. Redaction rectangles have to be defined, either by using pdfAutoSweep which uses regular expressions to search for matching text and coordinates, or by programmatically enterting the coordinates. pdfSweep will redact both text and images ensuring complete confidentiality.

Learn more at https://itextpdf.com/itext7/pdfSweep