Design Doc 0.1.0

FdfExtract

Minimalist Package to Extract Notes from FoxIt FDF Documents

PREPARED BY RICHARD J. CORDES
RJ.Cordes@COGSEC.org

PREPARED JUNE 2019

Contents

Executive Summary

The intent of this project was to develop a highly minimalist package to pull comments from FoxIt FDF documents and output a general-purpose object containing relevant data.

Introduction

Forms Data Format files are plain text files which generally contain *forms* data. However, they are in general use for most text file exports from PDFs (portable document format) and carry a large amount of unwieldly formatting data even in the case that the relevant material within the FDF is simple text. This package was not intended to be stand-alone and will only serve to extract comment information from FoxIt Notes, not FDF files in general.

Acronyms and Terms

For the purposes of this document, the following acronyms and terms have been clarified:

Term	**Definition**
DocLib	The document library, where PDF and similar documents are stored
FDF	Forms Data Format files. Intended for holding forms data, but are in general use for text files exported from PDFs
FCE	FoxIt Comment Export. A file of type FDF which is the result of an export of annotations from a FoxIt PDF. Holds formatting and other data as XML.
DTD	Document Type Definition. A definition of a document type as per SGML markup language

	standards. XML and HTML are examples of mark-up languages.
AO	Annotation Object. Defined within FCE. Contains comment text.
SPO	Source Path Object. Defined within FCE. Contains source path of the pdf the FCE was generated from.

Table 1: Terms and Acronyms

Requirements

This is a minimalist package; its only requirement was that it needed to be able to facilitate pulling notes and *some* meta-data from FoxIt Comment Exports (FCEs) which come in the form of FDFs, as previously noted.

Constraints and Concerns

There have been several items which are expected to impact and limit the design of the package. While each has the potential to be remedied by additional features, these limitations are beyond the scope of the project and must be considered.

- Document Meta-Data (author, title, publishing information) not included in the FDF.
- FDF has been found to occasionally include notes which have been deleted. It will include an annotation object, but no inner text. The reason for this has not been found and replication of the bug has not been successful. However, it is known to exist.
- Must be built with the assumption that there was no naming/declaration convention or Document Type Definition (DTD) used by the author of the notes.
- The structure of the incoming FDF is predefined and is not guaranteed to remain unchanged in its format

Package Overview

The package, being minimalist, wasn't expected to require any notable amount of code and was written in Golang.

Incoming FDF

While the format of the FCE is not *expected* to change, it is not guaranteed to remain stable nor is it known to have always been the format in use. That being the case, the relevant objects and structure take this into account.

Current FCEs begins with an initial heading:

```
%FDF-1.2
```

Code Snippet 1: FCE Heading

Beyond the heading, the FCE is composed of objects:

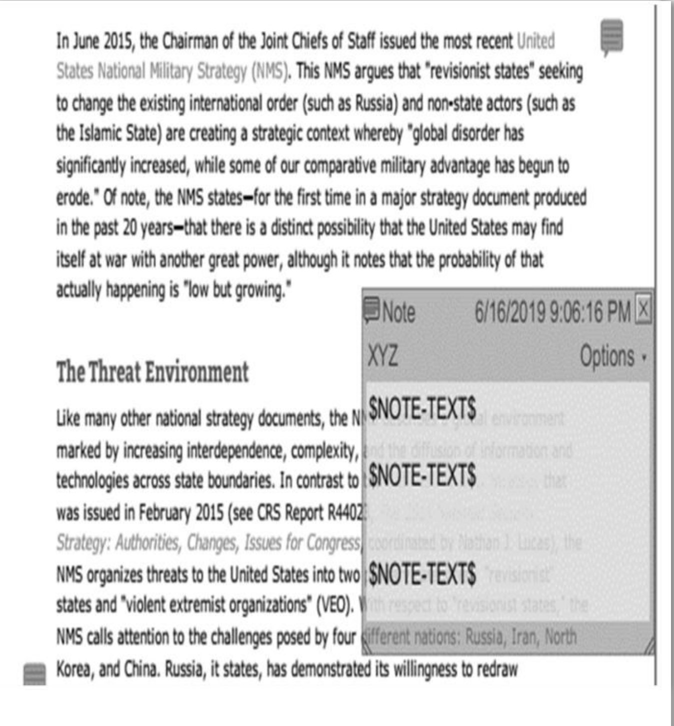
```
2 0 obj

<</C[ 1 1 0.192157]/Rect[ 561.375 355.75 581.375 375.75]/F 28/Subj(Note)/Name/Comment/Popup 3 0 R /M(D:20190616210616-
05'00')/CreationDate(D:20190616210525-05'00')/Page 0/RC(<?xml version="1.0"?><body xmlns="http://www.w3.org/1999/xhtml"
xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/" xfa:APIVersion="Acrobat:11.0.0" xfa:spec="2.0.2"><p dir="ltr"><span style="text-
align:left;font-size:13pt;font-style:normal;font-weight:normal;color:#000000;font-family:Arial">$NOTE-TEXT$&#x0A;</span></p><p
dir="ltr"><span style="text-align:left;font-size:13pt;font-style:normal;font-weight:normal;color:#000000;font-
family:Arial">&#x0A;</span></p><p dir="ltr"><span style="text-align:left;font-size:13pt;font-style:normal;font-
weight:normal;color:#000000;font-family:Arial">$NOTE-TEXT$&#x0A;</span></p><p dir="ltr"><span style="text-align:left;font-
size:13pt;font-style:normal;font-weight:normal;color:#000000;font-family:Arial">&#x0A;</span></p><p dir="ltr"><span style="text-
align:left;font-size:13pt;font-style:normal;font-weight:normal;color:#000000;font-family:Arial">&#x0A;</span></p><p dir="ltr"><span
style="text-align:left;font-size:13pt;font-style:normal;font-weight:normal;color:#000000;font-family:Arial">$NOTE-
TEXT$</span></p></body>)/T(XYZ)/Contents($NOTE-TEXT$\r\n\r\n$NOTE-TEXT$\r\n\r\n\r\n$NOTE-
TEXT$)/Subtype/Text/Type/Annot/Rotate 0/CA 1/NM(9f3bfd4a-81de-4949-96cc-e358afd70507)>>

endobj
```

Code Snippet 2: FCE Annotation Object (Relevant Data Emphasised)

As can be seen from the annotation object above, there is a large amount of formatting data that comes alongside simple note text. The note that this object is associated with only contains 33 characters. This accompanying formatting data helps not just to format the text within the note correctly, but also to format the note's container as well. The note that this object is associated with only contains 33 characters.

Luckily, regardless of how much the annotation object may change in the future, it is highly unlikely for FoxIt to cease using FDF, generally speaking. Therefore, in some form or another, it will be highly unlikely to deviate from XML's DTD format.

From each annotation object, all that needs to be retrieved is the inner comment text and the page number. This isn't too concerning, given that the document stores the former towards the top and the latter in full towards the bottom. In addition, it stores each item in a single line, and delimits objects using the x y obj and endobj tags.


In June 2015, the Chairman of the Joint Chiefs of Staff issued the most recent United States National Military Strategy (NMS). This NMS argues that "revisionist states" seeking to change the existing international order (such as Russia) and non-state actors (such as the Islamic State) are creating a strategic context whereby "global disorder has significantly increased, while some of our comparative military advantage has begun to erode." Of note, the NMS states—for the first time in a major strategy document produced in the past 20 years—that there is a distinct possibility that the United States may find itself at war with another great power, although it notes that the probability of that actually happening is "low but growing."

Note 6/16/2019 9:06:16 PM

XYZ Options

$NOTE-TEXT$

$NOTE-TEXT$

$NOTE-TEXT$

The Threat Environment

Like many other national strategy documents, the N
marked by increasing interdependence, complexity,
technologies across state boundaries. In contrast to
was issued in February 2015 (see CRS Report R4402
Strategy: Authorities, Changes, Issues for Congress,
NMS organizes threats to the United States into two
states and "violent extremist organizations" (VEO).
NMS calls attention to the challenges posed by four different nations: Russia, Iran, North
Korea, and China. Russia, it states, has demonstrated its willingness to redraw

Image 1: Annotation Example

Some light clean-up on the extracted text was required, but beyond this, there isn't much expected in terms of requirements for text formatting.

Architecture and Components

The structure is straightforward and is detailed in the following sections.

Exported

The user is provided with very few components.

Reader (Interface)

Reader is an interface which has only one exported method, Read(*path*). Read returns a CommentBlock interface to the user, which is detailed in the next section. Concurrency functions are currently disabled and are not necessary for the time being, however, they were included for testing purposes.

```
// A Reader takes a filepath to an FDF document and outputs a CommentBlock
type Reader interface {
	Read(string) (CommentBlock, error) // takes filepath and returns CommentBlock
	enableConcurrency()                // set FdfReader to be Concurrent
	disableConcurrency()               // set FdfReader to be not be Concurrent
}
```

Code Snippet 3: Reader (Interface)

CommentBlock (Interface)

CommentBlock is just an interface with two exported *getter* methods: one to get a list of Comment Interfaces and the other to get the source file.

```
// CommentBlock includes methods to get comments pulled from an FCE and return
// the path they were received from
type CommentBlock interface {
	GetSourceFile() []byte
	GetComments() []Comment
}
```

Code Snippet 4: CommentBlock (Interface)

Comment (Interface)

Comment is an interface with two getter functions, one to get text content and the other to get the page number the FoxIt comment it was found on.

```
// Comment has methods to pull annotation info found within a foxit document
type Comment interface {
	GetContent() []byte
	GetPageNumber() int
}
```

Unexported

The package has a current and tested version of an underlying reader, comment, and commentblock struct. The features of the package worth noting are detailed below.

lineParser (Func Type)

The lineParser func type allows for the creation of new parsers in the future, specific to doc headings that are currently unknown. At the moment, the only doc-heading specified is Fdf-12, which indicates the type of doc heading common to FoxIt fdf documents.

```
// LineParser functions takes bytes, parse them, and returns a Comment
type lineParser func([]byte) (Comment, error)
```

Code Snippet 5: lineParser (Func Type)

Use

Use of the package is quite simple. A reader is instantiated, reads from a file and the resulting CommentBlock can be used to pull the comments of interest.

```
func main() {
    reader := FdfExtract.NewReader()
    cb, err := reader.Read("AIE Automating Scientific… DARPA.fdf”)
    _ = cb.GetComments()
```

Code Snippet 6: Use Case

Current Status and Future Development

The Project will remain in “in-use” testing for around 1 month before being deployed as a 1.0.0 release in order to ensure stability and give the opportunity for diffusion, optimisation, and correcting of documentation. Concurrent reading within the reader has been removed prior to version 0.1.0 release, as it has been deemed to be unnecessary for the moment.

In the future, it is still expected that this may be expanded to include FDF annotation formats and that the objects will have methods to export to CSV and JSON, but most development should be found in packages which make use of FdfExtract.