# abbyy_to_epub3 Documentation

***Release 1.0***

**Deborah Kaplan**

**Nov 01, 2017**

# Contents:

CHAPTER 1

# ABBYY XML to EPUB3

## 1.1 Introduction

This module transforms ABBYY XML documents, generated by ABBYY FineReader 10, into primitively accessible ePub 3. The code is optimized for ABBYY XML documents created by the Internet Archive, though it may work for other ABBYY XML as well.

CHAPTER 2

# Features

1. Unicode-compliant
2. Can handle left-to-right and right-to-left text.
3. Attempts to recognize running headers, footers, and decimal or page numbers. Level of confidence in fuzzy matching can be fine tuned in `config.ini`. Errs on the side of minimizing false positives.

CHAPTER 3

# Limitations

1. Accessibility is inherently limited by the input ABBYY FineReader documents. If they are marked up with headings and other semantic markup, that structure will be incorporated into the ePub.
2. There is currently no functionality for image description.
3. The module can also transform ABBYY XML documents generated by ABBYY FineReader 6. However, those documents are not marked up with headings, so there is no structural navigation for accessibility.

CHAPTER 4

# Requirements

- Python 3
- If running epubcheck, a Java Runtime environment
- If running DAISY Ace, Node.js

CHAPTER 5

# Usage

From within a Python program:

```
from abbyy_to_epub3 import create_epub
book = create_epub.Ebook('docname')  # See *Assumptions* below.
book.craft_epub()
```

From the shell:

```
abbyy2epub docname      # See *Assumptions* below.
```

The available command line arguments are:

..code:: bash

usage: abbyy2epub [-h] [-d] [–epubcheck] [–ace] docname

Process an ABBYY file into an EPUB

**positional arguments:**

**docname A directory containing all the necessary files. See the README** for details.

**optional arguments:**

| | |
|---|---|
| **-h, --help** | show this help message and exit |
| **-d, --debug** | Show debugging information |
| **--epubcheck** | Run EpubCheck on the newly created EPUB |
| **--ace** | Run DAISY Ace on the newly created EPUB |

CHAPTER 6

# System dependencies

If you'd like to run epubcheck, there are certain system dependencies. Depending on running environment, these may need to be manually installed. On Ubuntu, I installed these with:

```
sudo apt-get install default-jre libpython3-dev
```

If you'd like to run the DAISY Ace accessibility checker, you'll also need Node.js and Ace. On Ubuntu, I installed these with:

```
sudo apt-get install nodejs
sudo npm install ace-core -g
```

If Ace successfully installed, you should be able to run:

```
ace --help
```

at the command line. This should display usage information. For more information see the *Ace Getting Started Guide <http://inclusivepublishing.org/toolbox/accessibility-checker/getting-started/>*.

CHAPTER 7

# Installation

This package can be installed on your local system. From the directory containing setup.py:

```
pip install -r requirements.txt
python setup.py develop
pip install .
```

You can rebuild the documentation, which is generated with Sphinx.

```
cd docs
make html
```

CHAPTER 8

# Testing

Run `py.test` from the top-level app directory. Create new tests in the `tests` subdirectory.

CHAPTER 9

# Assumptions

This application assumes you are working in a directory which contains a subdirectory for the document and a specific set of files. If the document is named `docname`, the directory structure assumed is:

```
docname/
    docname_abbyy.gz
    docname_meta.xml
    docname_jp2.zip
```

- `docname_abbyy.gz` unzips to `docname_abbyy`, an XML file generated by ABBYY.
- `docname_jp2.zip` unzips to a directory called `docname_jp2`, which includes a number of documents in the format `docname_####.jp2`.
  - `docname_0000.jp2` is scanner calibration.
  - `docname_0001.jp2` is the cover image and the first image reference in the ABBYY.

# CHAPTER 10

# Contribute

- Source code on GitHub
- Issue tracker

## 10.1 abbyy_to_epub3

### 10.1.1 abbyy_to_epub3 package

#### Submodules

#### abbyy_to_epub3.constants module

#### abbyy_to_epub3.create_epub module

#### abbyy_to_epub3.parse_abbyy module

**class** abbyy_to_epub3.parse_abbyy.**AbbyyParser**(*document*, *metadata_file*, *metadata*, *paragraphs*, *blocks*, *debug=False*)

Bases: object

The ABBYY parser object. Parses ABBYY metadata in preparation for import into an EPUB 3 document.

Here are the components of the ABBYY schema we use:

```
<page>
    <block>types Picture, Separator, Table, or Text</block>
```

Text:

```
<page>
        <region>
        <text> contains a '\n' as a text element
```

```
        <par> The paragraph, repeatable
            <line> The line, repeatable
                <formatting>
                <charParams>: The individual character
```

Image: Separator: Table:

```
<row>
  <cell>
    <text>
      <par>
```

Each paragraph has an identifier, which has a unique style, including the paragraph's role, eg:

```
 <paragraphStyle
     id="{000000DD-016F-0A36-032F-EEBBD9B8571E}"
     name="Heading #1|1"
     mainFontStyleId="{000000DE-016F-0A37-032F-176E5F6405F5}"
     role="heading"
     roleLevel="1"
     align="Right"
     startIndent="0" leftIndent="0"
     rightIndent="0" lineSpacing="1790" fixedLineSpacing="1">
<par align="Right" lineSpacing="1790"
     style="{000000DD-016F-0A36-032F-EEBBD9B8571E}">
```

The roles map as follows:

| Role name | role |
|---|---|
| Body text | text |
| Footnote | footnote |
| Header or footer | rt |
| Heading | heading |
| Other | other |
| Table caption | tableCaption |
| Table of contents | contents |

**etree = ''**

**is_block_type**(*elem*, *blocktype*)
Identifies if an XML element is a textblock.

**ns = ''**

**nsm = ''**

**parse_abbyy**()
read the ABBYY file into an lxml etree

**parse_content**()
Parse each page of the book.

**parse_metadata**()
Parse out the metadata from the _meta.xml file

**parse_paragraph_styles**()
Paragraph styles are on their own at the start of the ABBYY

**version = ''**

abbyy_to_epub3.parse_abbyy.**add_last_text**(*blocks*, *page*)
Given a list of blocks and the page number of the last page in the list, mark up the last text block for that page in the list, if it exists.

abbyy_to_epub3.parse_abbyy.**gettext**(*elem*)

### abbyy_to_epub3.utils module

abbyy_to_epub3.utils.**dirtify_xml**(*text*)
Re-adds forbidden entities to any XML string. Could cause problems in the unlikely event the string literally should be '&amp'

abbyy_to_epub3.utils.**is_increasing**(*l*)
Given a list, return True if the list elements are monotonically increasing, and False otherwise.

abbyy_to_epub3.utils.**sanitize_xml**(*text*)
Removes forbidden entities from any XML string

### Module contents

CHAPTER 11

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## a

# Index