code{4}lib JOURNAL

Issue 24, 2014-04-16 ISSN 1940-5758

EPUB as Publication Format in Open Access Journals: Tools and Workflow

In this article, we present a case study of how the main publishing format of an Open Access journal was changed from PDF to EPUB by designing a new workflow using JATS as the basic XML source format. We state the reasons and discuss advantages for doing this, how we did it, and the costs of changing an established Microsoft Word workflow. As an example, we use one typical sociology article with tables, illustrations and references. We then follow the article from JATS markup through different transformations resulting in XHTML, EPUB and MOBI versions. In the end, we put everything together in an automated XProc pipeline. The process has been developed on free and open source tools, and we describe and evaluate these tools in the article. The workflow is suitable for non-professional publishers, and all code is attached and free for reuse by others.

by Trude Eikebrokk, Tor Arne Dahl and Siri Kessel, Oslo and Akershus University College of Applied Sciences; with thanks to: Eirik Hanssen

Introduction

Oslo and Akershus University College of Applied Sciences (HiOA) offers a publishing platform based on the open source software Open Journal Systems (OJS). The Learning Centre and Library maintains the platform.

After a couple of years working with Open Access publishing it has become apparent that many online journals are still based on a print-centric publication model. It is easy to establish an e-journal using a traditional print workflow in an academic environment, because most researchers use a word processor as their major work tool. Many know a bit about copy-editing from their contact with scholarly journals, and current word processors can save documents as PDF files. That is why PDF is the standard format used in the Open Access journals appearing outside the professional publishing industry.

We decided to establish a project with the goal of creating a new workflow for the journals using EPUB as the main publication format. The project succeeded, and our aim is to share experiences and guide others planning to do the same.

Why not PDF?

There are two important reasons why we wanted to replace PDF as our primary e-journal format.

Device independency

We started considering alternatives to PDF after trying to read some of the journal articles on e-book readers (a Sony Reader Touch PRS-650 and a PaperCaster Boox). PDF files do not work well on these E Ink devices. There are font problems and it is hard to scale the text size. There are solutions to some of these problems, but PDF is a print format. It will never be the best choice for reading on tablets (e.g. iPad) or smartphones, and it is challenging to read PDF files on e-book readers with E Ink displays (like most Amazon Kindle models and the Sony Reader). We wanted to replace or supplement the PDF format with EPUB to better support digital reading. This means that we needed to change the entire workflow in the journal in order to take full advantage of the opportunities provided by digital publishing.

Universal design and accessibility

Our second reason for replacing PDF with EPUB was to alleviate accessibility challenges. PDF is a format that can cause many barriers, especially for users of screen readers (synthetic speech or braille). For example, Excel tables are converted into images, which makes it impossible for screen readers to access the table content. PDF documents might also lack search and navigation support, due to either security restrictions, a lack of coded structure in text formats, or the use of PDF image formats. This can make it difficult for any reader to use the document effectively and impossible for screen reader users. On the other hand, correct use of XHTML markup and CSS style sheets in an EPUB file will result in search and navigation functionalities, support for text-to-speech/braille and speech recognition technologies.

In the last decade, accessibility and universal design have become a legal issue, both nationally and internationally. Access to web content is required through article 9 in the UN convention on the rights of persons with disabilities (CRPD) [1] and some national anti-discrimination legislation and procurement regulations. The Norwegian Discrimination and Accessibility Act (DAA) [2] authorizes universal design of ICT in articles 13 and 14, and associated regulations for ICT will come into force July 1, 2014. DAA defines universal design as “can be used by as many people as possible”.

Accessibility is therefore an essential aspect of publishing e-journals: we must consider diverse user perspectives and make universal design a part of the publishing process.

As early as in 1996, the World Wide Web Consortium (W3C) established the Web Accessibility Initiative (WAI). WAI has made technical guidelines for accessibility for all, e.g. to web content [3][4] and authoring tools [5][6]. The Norwegian standardization organization (Standards Norway) released a standard for electronic documents in 2013. These guidelines and standards are useful tools when establishing a new publication workflow and format for e-journals.

It is difficult to apply the WAI guidelines and make the e-journals fully accessible within the current workflow of e-journal publishing at HiOA. Some of the main requirements for universally designed e-journals that can be satisfied by the EPUB format are:

- text format
- compatibility with different devices, software and older versions of devices and software
- structured markup of titles, headings, links, notes, tables, etc.
- informative alternative text on graphs and essential images
- layout with style sheets
- liquid layout of font size and line space, enlarging that adjust to the screen size
- high contrast between text and background
- highlighting of text or items in at least two ways

Why EPUB?

EPUB solves both of the problems mentioned above. EPUB is a reflowable format. This means that the text will always fit the screen without the need for horizontal scrolling. The user can increase and decrease the font size without any changes in page width. This accessibility quality has always been a design goal of the W3C web standards. EPUB is based on web standards and inherits these qualities.

Additionally, EPUB is an open standard maintained by the International Digital Publishing Forum (IDPF). On the other hand, Amazon's file format AZW/MOBI is a proprietary format. Amazon Kindle devices cannot read EPUB files directly, but free tools like Calibre can easily convert an EPUB file to the native Amazon Kindle format.

EPUB has other advantages as well. The current version, EPUB 3 from October 2011, is based on a subset of HTML5 and CSS3, making it more suitable for multimedia content than earlier versions. At present, few reading devices and applications support EPUB 3.

EPUB 2 from 2007 is still the most widely used and supported version of the format and works well for most academic journals. That is why we will concentrate on EPUB 2 in this article.

The case

Our goal was to change the publication format in *Professions & Professionalism* from PDF to EPUB. *Professions & Professionalism* is an Open Access journal that "invites research-based empirical, theoretical or synoptic articles focusing on traditional professions as well as other knowledge-based occupational groups approached from any perspective or discipline" [7]. The first issue was published in November 2011. The articles have CC-BY licences and are typically text-based with references, footnotes and tables. Some contain images, too.

The current workflow follows these steps:

1. The authors upload their articles to the OJS software. Most articles are uploaded as Microsoft Word documents (meaning the DOCX format).
2. The editors distribute these Microsoft Word documents for peer-review and send the comments to the authors.
3. The authors update their Microsoft Word documents according to the peer reviewers' comments and re-upload them into OJS.
4. The copy-editor does the finishing work on layout, proofreading and reference-checking. When finished, a document is saved as a PDF file.
5. The editor sends the PDF file by e-mail to the author, who will then check the final layout.
6. If there are changes, the copy-editor will apply the changes and save the final PDF document that is the published online version of the article.

First attempt: Conversion tools

At first, we thought this was going to be a very simple project requiring little effort. We would apply some of the easy-to-use DOC to EPUB converters freely available and end up with well-formatted and functional EPUBs ready for publishing. Our prediction was that we maybe would have to ask the journal managers to tweak a thing or two in their original Microsoft Word files, but under no circumstances did we expect to have to make any big changes in the journal's existing method of editorial work.

We researched what kinds of tools were available for EPUB conversion. The following table describes the tools we tested and some of the issues we encountered. To validate the results, we used EpubCheck.

Software	Description	Conversion	Main issues
Calibre	An open source e-book library management application that can convert e-books to different formats.	Calibre is unable to convert directly from DOCX (Microsoft Word), but can convert the Open Document format (ODT). There are tools for DOCS to ODT conversion, for example by way of OpenOffice or LibreOffice.	Broken tables, separated image and captions, footnotes at end of article counting only to 9. The remaining footnotes had number 1. Missing headers and footers. EPUB file not valid.
Sigil	A multi-platform EPUB ebook editor and open source software.	Sigil converts the DOCX file to filtered XHTML and packs it as an EPUB file.	EPUB file not valid.
Writer2epub	An OpenOffice.org extension that creates an EPUB files from the word processor.	Install the Open Office extension. Open the document; choose Writer2epub to generate the EPUB file. Mainly a voluntary effort.	Images missing. EPUB file not valid.
Adobe InDesign	Adobe InDesign has native support for EPUB export.	Create or open a document in Adobe InDesign. Export the document to EPUB.	EPUB file not valid. None of our journals actually uses InDesign in their publishing process, so that would mean another tool for them to learn. Also licensed, so they would have to pay for it.

None of the resulting EPUB files were valid. Because non-valid EPUB files give unpredictable results on different reading devices (with some e-readers even rejecting such files), none of the tools we tested seemed conducive to accessibility in our academic e-journals.

New workflow

Since none of the existing conversion tools were sufficient, we decided to change direction and start looking into developing our own conversion workflow. We considered marking up the journal articles directly as EPUB, but discarded this approach because EPUB, like XHTML, is more of a presentation format than a rich semantic publishing format. For instance, footnotes and references are typical parts of a journal article, but there are no such elements in XHTML.

Because an EPUB file is a collection of XML files, creating our own XML-based process to generate EPUB files seemed possible. We also saw XML as an ideal format for long-term preservation of the articles: it is open (not proprietary), can be read by humans and machines and is software independent.

Input: The Journal Article Tag Suite (JATS)

Public Library of Science (PLOS) is the flagship of Open Access publishing, currently publishing seven peer-reviewed Open Access journals. The main publishing format in the PLOS journals is the native web format (in this case XHTML 1.0 Transitional) with lots of hypertext links, for instance in the references section (for an example article, see Michel and Knouft [8]). PDF and—most interesting for us—XML are alternative formats.

PLOS uses Journal Publishing Tag Set Version 3.0 as its XML source format. The National Center for Biotechnology Information (NCBI), which is part of the U.S. National Library of Medicine (NLM), develops and maintains this XML application. The Journal Publishing Tag Set is part of a larger family of XML applications called NLM Journal Archiving and Interchange Tag Suite. PubMed Central, the free archive of biomedical and life sciences journal literature at the U.S. Library of Medicine, requires use of the tag set in their file submission specification [9].

When we started working on our project, we found out that the tag suite was being prepared for a National information Standards Organization (NISO) standardization process. NISO formally approved the standard on 9 August 2012, and it is referred to as Journal Article Tag Suite (JATS), version 1.0 (ANSI/NISO Z39.96-2012). This is the version we have used in markup and will refer to from now on in the text.

Output: EPUB

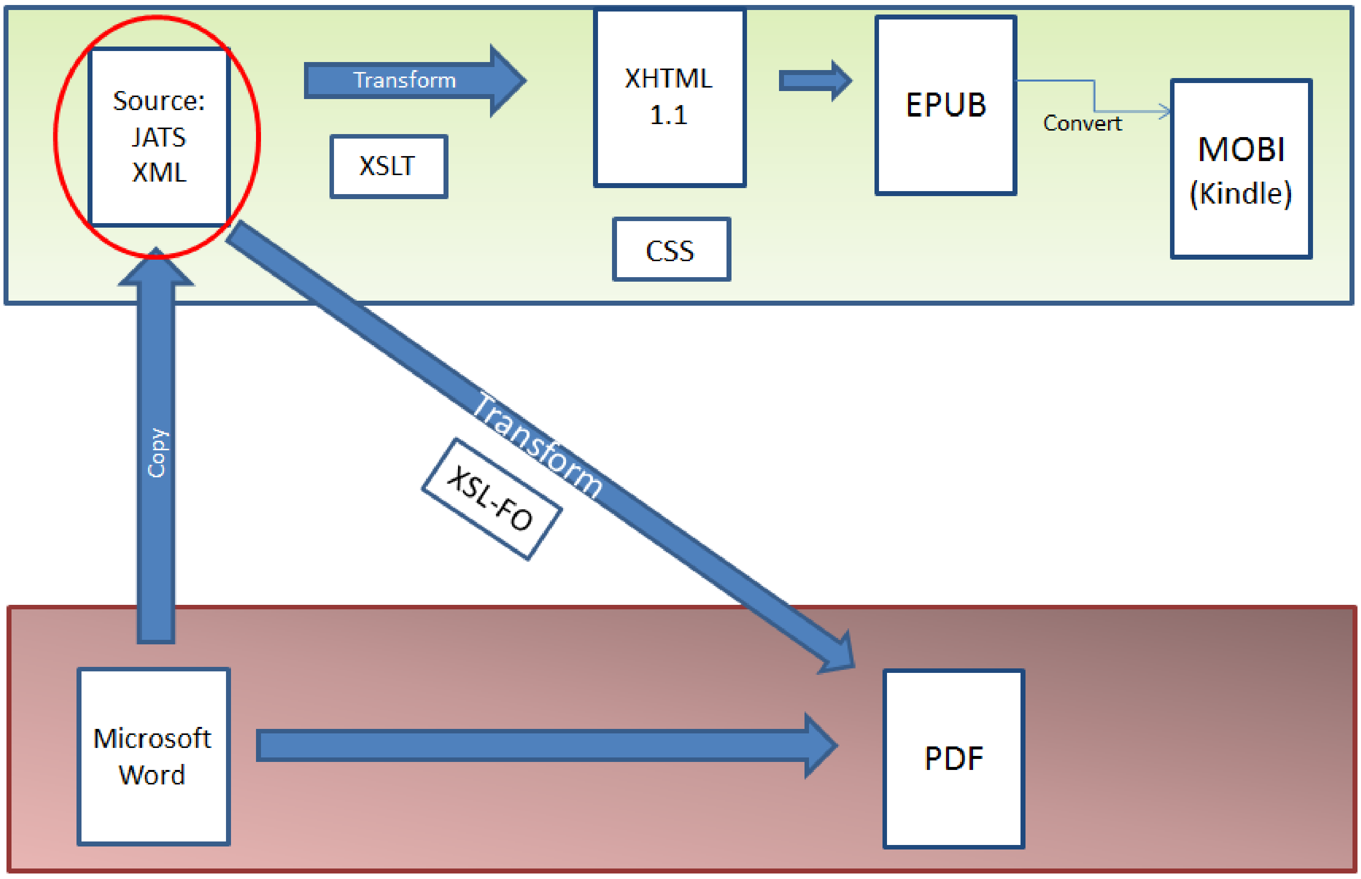
EPUB 2 consists of three sub specifications: the Open Publication Structure (OPS), the Open Packaging Format (OPF) and the Open Container Format (OCF) [10]. The resulting EPUB 2 file is a zipped archive of a simple file structure including some required files with metadata information about the files in the archive. A typical and simple EPUB file has this content:

- A root folder with one file called `mimetype`. This file always has this content: `application/epub+zip`.
- A subfolder called `OEBPS` (an acronym for Open eBook Publication Structure, which is inherited from EPUB's predecessor file format). The subfolder has the following content:
  - All the files with the main content of the EPUB 2 file, i.e. XHTML for structure and content, CSS for presentation, and any pertaining image files. These files can be organized in subfolders or put directly into the `OEBPS` folder.
  - The EPUB "root file", which is an XML file with the suffix `.opf`. The file has a metadata part (described in Dublin Core), a manifest part (a list of all the files in the EPUB zip file), and a spine part (defining the linear reading order of the files in the manifest part). The complete structure of this file is described in the OPF specification.
  - A navigation control XML file (with the suffix `.ncx`) that works like a table of contents and will help the user navigate through the content.
- A subfolder called `META-INF` with only one file called `container.xml` (this is the only required file in the folder according to the EPUB 2 specification). The XML file has a reference to the name of a required OPF file (the EPUB "root file") located in the `OEBPS` folder.

Overview

The figure below illustrates the old and the new workflow. The lower part of the figure shows the old workflow, where we simply save the Microsoft Word articles as PDF and publish them online.

The upper part of the figure illustrates the new workflow. We mark up the content of the Microsoft Word document from the old workflow as a JATS XML document, and that is the only manual work to do. We generate the different publishing formats—XHTML, EPUB and MOBI—by running a script which is attached to this article.



The National Center for Biotechnology Information (NCBI) has created a toolset of XSLT style sheets for transforming JATS documents to (X)HTML and XSL-FO. We have only implemented the EPUB workflow so far but will create PDF versions by way of the XSL-FO style sheet in the future.

Building the EPUB file is not part of the NCBI toolset, so that is our contribution to the production pipeline. We did this by adding XSLT style sheets transforming the JATS source file into the mandatory files in the EPUB file structure explained above.

Our goal was to automate the EPUB production process completely. The NCBI toolset provides two shells or pipelines, putting all the steps together. One of them uses proprietary Saxon extensions and the other uses XProc, which is a W3C recommendation.

In the following sections, we go through the steps in the new workflow used in *Professions & Professionalism* to replace the PDF format with three digitally native formats: XHTML, EPUB and MOBI. To illustrate, we use the article *Medical management in Norwegian hospitals* by Ivan Spehar and Lars Erik Kjekshus [11] from *Professions & Professionalism*.

The article contains 4 tables, 2 illustrations and 56 references of different kinds (books, chapters in anthologies, journal articles, reports and PhD theses). We chose the example article because of its complexity, which should provide others with multiple templates for XML markup.

JATS markup

As mentioned, we chose JATS 1.0 as the native XML application in the new workflow. All XML files, including the example article, conform to this standard (or more specifically the Journal Publishing Tag Set that is a part of JATS).

When we start JATS markup, we have the final Microsoft Word version of the article. We consider it unrealistic and unnecessary to use XML throughout the work cycle. Microsoft Word is the daily writing tool for the article authors and the copy-editor, and we do not want to change their already efficient collaboration and communication.

We copy and paste the text from the Microsoft Word document to the content of the root element of a JATS document in an XML editor. There are many XML editors to choose from. Some are parts of integrated and commercial software packages, like Liquid XML Studio, Oxygen XML Editor and XMLSpy, but we chose the simpler XML Copy Editor, which is free software released under the GNU General Public License (GPL) version 2.

XML Copy Editor has the basic features that we needed to do most of the markup and transformations: XML editor, validation against DTDs and XML Schemas, and XSLT transformations. It only supports XSLT and XPath version 1.0, but that works in most cases.

The example JATS document has three parts, which are child elements to the root element:

- `head`: Metadata about the article, including title, authors and their affiliations, classification and subject headings, license and the abstract.
- `body`: The main content of the article, divided into sections and paragraphs.
- `back`: This is the reference section.

The markup of the `head` part is not very challenging, but the journal has to have some clear principles about classification and subject headings to create a taxonomy that clusters articles about the same topics together. This will help navigation on the website as the number of archived issues increases.

We use a standard CC-BY license on all our articles. For this, JATS has a `permissions` element, which is a child to the `head` element:

```
<permissions>
  <license license-type="open-access"
xlink:href="http://creativecommons.org/licenses/by/3.0/">
     <license-p>This is an open-access article distributed under the terms
of the Creative Commons Attribution License, which permits
unrestricted use, distribution, and reproduction in any medium,
provided the original work is properly cited.</license-p>
  </license>
</permissions>
```

The `body` part is straightforward markup for anyone familiar with HTML. Some elements have different names in JATS (for example, `italic` for `i`, `bold` for `b`), but the semantics are the same. The hierarchical structure of the article is marked up using the `sec` element (which uses semantics identical to `section` in HTML5). Sections can be nested into any number of subsection levels. Each (sub)section has a `title`, which is the heading. Paragraphs are marked up using the `p` element, just as in HTML. Manual markup of tables is time-consuming, but not difficult. However, we emphasize the need for correct markup of row and column headings for accessibility reasons.

The most complex markup is the reference section in the `back` part of the JATS document. The references can either be marked up using the `element-citation` element, or the `mixed-citation` element. The difference is that `element-citation` is a strictly structural markup of the bibliographic description of the work, while `mixed-citation` is a simpler, more presentational markup of it. Using `mixed-citation`, spacing and punctuation are preserved during transformation to other formats like HTML or PDF. With `element-citation`, the XSLT style sheet is responsible for spacing, punctuation and italicization.

We chose to use `element-citation` because of its rich semantics. The structural markup takes more effort than `mixed-citation`, where just some parts of the description are marked up, but the richer semantics provide more flexibility.

The examples below show the differences between the two markup styles on a bibliographic description of an online journal article. First is the `element-citation` version that we have used:

```
<ref id="AW2008">
  <element-citation publication-type="journal"
publication-format="online">
     <person-group person-group-type="author">
        <name>
           <surname>Album</surname>
           <given-names>D.</given-names>
        </name>
        <name>
           <surname>Westin</surname>
           <given-names>S.</given-names>
        </name>
     </person-group>
     <year>2008</year>
     <article-title>Do diseases have a prestige hierarchy? A survey among
physicians and medical students</article-title>
     <source>Social Science &amp; Medicine</source>
     <volume>66</volume>
     <issue>1</issue>
     <fpage>182</fpage>
     <lpage>188</lpage>
     <uri>http://dx.doi.org/10.1016/j.socscimed.2007.07.003</uri>
  </element-citation>
</ref>
```

As can be seen, every part of the bibliographic description is marked up with elements. On the other hand, a `mixed-citation` has mixed content with only some parts marked up. The dots, commas, brackets, and other punctuation are preserved through transformations. This can be an advantage when dealing with uncommon publication types.

```
<ref id="AW2008">
  <mixed-citation publication-type="journal">Album, D., &amp; Westin, S.
(2008). Do diseases have a prestige hierarchy? A survey among physicians
and medical students. <source>Social Science &amp; Medicine</source>,
<volume>66</volume>(1), 182-188.
  <uri>http://dx.doi.org/10.1016/j.socscimed.2007.07.003</uri>
  </mixed-citation>
<ref>
```

In both cases, the citation has the parent element `ref`, which has the mandatory attribute `id`. The value of the `id` attribute must be a unique identifier for the reference in the XML document. This identifier is used as a cross-reference when citing the work in the article. JATS has the `xref` element for this. The `xref` element can take several attributes, but we only use two of them:

- `ref-type`: The type of reference. For bibliographic references, `bibr` is the correct value.
- `rid`: The cross-reference to the unique identifier in the reference list. If the value of the attribute is not in the reference list, the XML document will not be valid.

When citing the reference in the example, the XML markup looks like this:

```
<xref ref-type="bibr" rid="AW2008">Album &amp; Westin, 2008</xref>
```

The unique identifiers are very important, but the person doing the markup must make them up. We found a system for creating them that simplifies markup and ensures the quality of the reference list. We combine the first letters of the authors' surnames with the publishing year, and then, if the combination is not unique, we add a letter (a, b, c etc.). In the example, the identifier is **AW2008** from **A**lbum, D., & **W**estin, S. (**2008**). As mentioned, the JATS document will not be valid if the `rid` attribute value of an `xref` does not match the identifier in the reference list. We have found several errors in the published articles this way.

Transformation to XHTML

The process of transforming a JATS document to an XHTML document using the NCBI toolset has three steps:

1. Pre-process the reference list. The toolset has XSLT style sheets for two citation styles, American Psychological Association (APA) and NLM/PubMed Central guidelines.
2. Transform the result document from step 1 to HTML.

3. Post-process the HTML document from step 2 to XHTML.

In our case, we have to go through all these steps, since our main goal is to produce EPUB files, and EPUB requires XHTML 1.1 for the main content file. The journals use APA as citation style, so we chose APA transformation as the base XSLT style sheet for references in the pipeline.

The main XSLT style sheet is `jats-html.xsl`, the web preview style sheet. There is also a CSS style sheet for the web presentation, `jats-preview.css`. None of them fit all our needs, so we had to customize them, just as we had to do with the APA pre-processing style sheet. We had to recreate the look-and-feel of the journal on the web and in the EPUB version.

We have used a trial-and-error approach in the project. We succeeded in the end, eventually realizing that merging changes directly into the NCBI style sheets is not the proper way of doing it. Piez calls this the 'monolithic' customization method [12]. All custom changes will be lost when upgrading the toolset. In his article, Piez recommends 'vertical customization', which means creating separate XSLT and CSS style sheets with custom templates and styles and then importing the NCBI master style sheets into them. The custom templates and styles will then override the master when needed, otherwise falling back on the master style sheet. In XSLT, the `xsl:import` instruction does the trick; in CSS the `@import` rule achieves the same.

Our custom style sheets—`hioa-citations-prep-APAcit.xsl`, `hioa-xhtml.xsl` and `hioa-web.css`—are provided with the article. The particular adjustments we have made will probably be of no use to others, but we would like to stress the importance of creating a vertical customization from the start. It will save you hours of work compared to separating them from the master style sheets afterwards.

It is also worth mentioning that the reference pre-processing style sheets from NCBI are using XSLT 2.0, which XML Copy Editor does not support. To run these transformations in tests before the final XProc pipeline, we had to download an XSLT 2.0 transformation engine. We chose Saxon-HE.

Packaging the EPUB file

The file structure of an EPUB file is described in the previous section titled Output: EPUB. All the mandatory files in the file structure are either static files or results of XSLT transformations of the JATS source file.

The XHTML version of the article is the output from the transformation described in the previous section. We use our customized CSS style sheet for presentation. These and any image files (GIF, JPEG, PNG) must be copied to the OEBPS folder or subfolders. `mimetype` and `container.xml` are static files. Only two files, `content.opf` (the EPUB root file) and `toc.ncx` (the table of contents) depend on the content of the particular EPUB file.

We can generate both of these files from the source JATS file using XSLT. We provide two style sheets: `epub-content.opf.xsl` for the EPUB root file and `epub-toc.ncx.xsl` for the table of contents file. They should be applicable for any other journal using the same process as we are describing here. The XSLT style sheet fetches the variable file content (title, author, metadata etc.) from the JATS source file. You should, however, rename the CSS style sheet from `hioa-epub.css` in `epub-content.opf.xsl` to the proper name used in the actual journal.

We tried using different zip tools to package the EPUB file but had problems with adding files in the correct order (the `mimetype` file must be first in the archive). However, the open source Windows software Info-ZIP worked well.

Conversion to MOBI

Amazon Kindle is the market-leading e-reader device but has no support for EPUB. The Kindle's native format is an Amazon proprietary format, but it also supports MOBI, which was developed by the French company Mobipocket that Amazon bought in 2005.

During the testing phase, we used Calibre to convert from EPUB to MOBI. The result is lossless and close to impeccable in all our test cases.

Putting it all together: The XProc pipeline

As mentioned above, the NCBI toolset includes a default XProc pipline, but we had to make several changes to make it suit our needs.

We provide the resulting file, `process-jats-xml.xpl`, as part of the source code. The pipeline uses an input folder (for JATS source files), a working directory for temporary files, and an output folder for the final result files.

Here is an overview of the pipeline:

1. Pre-process the JATS references to APA citation style with `hioa-citations-prep-APAcit.xsl`. We have made some slight adjustments to the XSLT style sheet `jats-APAcit.xsl`.

2. Fix some empty `id` attribute problems on graphic elements in the result file from the previous step. (These problems can probably be solved in a better way, but we have not yet managed to find a better way.)
3. Run the main transformation of the result file from step 2 with `hioa-xhtml.xsl`. This is our vertical customization of the master style sheet `jats-html.xsl`. In this case, we made major revisions to the master style sheet and chose to cast the result directly into XHTML rather than run the previously mentioned post-processing style sheet.
4. Fix empty namespaces in the result document from step 3. Just like step 2, this should ideally be a non-issue, but we have not solved it completely. Namespaces in XML generally and JATS specifically are challenging [13]. Most web browsers ignore empty namespaces, but in the strict XHTML 1.1 required by the EPUB standard, they cause crucial problems.
5. Transform the JATS source file into `content.opf` using `epub-content.opf.xsl`.
6. Transform the JATS source file into `toc.ncx` using `epub-toc.ncx.xsl`.

The XProc file has to run on an XProc engine. We used XML Calabash, developed by Norman Walsh. XML Calabash includes Saxon-HE for XSLT 2.0 transformations.

XProc is a vital part of the EPUB production but does not make up the full workflow. For example, an EPUB file is a zip-archive of the EPUB file structure, but we also need a MOBI version of this file. Therefore, we have created two scripts (a Windows batch file and a Linux Bash shell script) that include the XProc pipeline and also automate the rest of the process. The scripts take a JATS input file and generate three versions of the article: XHTML (for the web), EPUB and MOBI.

See the Source code section of this article for detailed information about downloading and running either script.

Costs

The main issue with our new workflow is that it is time consuming. The time we spent on each article depends on the length and complexity of the article, and it is hard to predict how long it will take to mark up each issue of the journal.

As a proof-of-concept we marked up all five articles in volume 2, issue 1 of *Professions & Professionalism* and measured the time spent on each article. We used XML Copy Editor for the manual markup and we received the articles in Microsoft Word format from the copy-editor of the journal. The articles had been through the editorial process and adjusted to a print layout. Much of this work, like hyphenation and APA formatting of the reference lists, will be unnecessary in the future, when we will use the author's postprints as the basis for markup.

One of the XML coders has a great deal of experience with XML; the other is an expert. The table below shows the time used on markup, and the articles are linked so you can see the level of complexity of each article. Tor Arne marked up Abrahamsen, Holte, & Laine, 2012, and Nygaard, 2012; while Trude marked up Saks, 2012, Kallberg, 2012, and Spehar & Kjekshus, 2012.

Article	Time (in hours)
Saks, 2012	3 ½
Nygaard, 2012	4 ¾
Kallberg, 2012	2 ½
Spehar and Kjekshus, 2012	2 ¼
Abrahamsen, Holte, and Laine, 2012	8 ¾
Sum	21 ¾

To compare our efforts, we asked the copy-editor to estimate the time he had spent on the copy-editing work. As mentioned in the section The case, Microsoft Word is used in the process. The copy-editor told us that it was hard to separate the different parts of his work but estimated that he had used "a full workday" on the issue, i.e. 7 ½ hours. He needed additional help from the journal editor on tables. The tables were exported graphic files from Microsoft Excel. This work took about 1 ½ hours, which means that the editorial work for the issue in the current production takes 9 hours. The XML markup takes a bit more than twice the time, but this additional work enables publishing in four additional formats: XML, XHTML, EPUB and MOBI.

Conclusion

In this project, we have established a new workflow for an e-journal to replace PDF with EPUB as the main publication format. By doing this we we have made the journal more accessible for everyone, both in terms of reading devices and user diversity. We have focused on accessibility, universal design, device and software independency, and preservability.

Marking up an article in XML is not very complicated and is something anyone should be able to do. As long as a document is correctly marked up as JATS XML, a single command is enough to run the entire XProc pipeline, which applies all the transformations in the correct order and creates output files in XHTML, EPUB and MOBI. The source code provided should make it easy for others to do the same.

An important outcome has been to convince the journal editors of the benefits of the changed workflow. An unforeseen aspect of this project has been the automated crosschecking of the references in the article. Any citation in the text that does not have a corresponding part in the reference list will stop the validation process. In addition, if any parts of the reference are missing (for example the value for issue) it will become very apparent in the markup process, as it will be an empty field in the markup. The editors consider this important and believe it adds value to the editorial process.

Our method will no doubt cost more in effort than the traditional Microsoft Word to PDF method that the journals so far have employed. However, we believe the value of the output we create will make up for the added costs.

We plan to implement the same workflow in our other journals in the future.

Source code

Readers can download a zip file from GitHub, which includes everything needed to run the pipeline with the example article as input.

The file is located at https://github.com/eirikhanssen/jats2epub and is licensed under a GPLv3 license.

Save the zip file to a folder and unpack the archive.

The distribution requires Java Runtime Environment but includes all the software needed with the exception of tools for EPUB to MOBI conversion. The script needs Amazon's KindleGen to do this. You can freely download the software from Amazon, but it is illegal to distribute it as part of the zip file.

The `readme` files in the zip archive document the distributed files, folders and usage.

Eirik Hanssen, Learning Centre and Library, Oslo and Akershus University College of Applied Sciences has written the scripts, the XProc file and the documentation.

References

[1] United Nations. 2006. Convention on the rights of persons with disabilities. Available from: http://www.un.org/disabilities/convention/conventionfull.shtml
[2] Barne-, likestillings- og inkluderingsdepartementet. 2008. Diskriminerings- og tilgjengelighetsloven. Available from: http://lovdata.no/dokument/NL/lov/2013-06-21-61
[3] W3C. 1999. Web Content Accessibility Guidelines 1.0. Available from: http://www.w3.org/TR/WCAG10/
[4] W3C. 2008. Web Content Accessibility Guidelines (WCAG) 2.0. Available from: http://www.w3.org/TR/WCAG20/
[5] W3C. 2000. Authoring Tool Accessibility Guidelines 1.0. Available from: http://www.w3.org/TR/ATAG10/
[6] W3C. 2013. Authoring Tool Accessibility Guidelines (ATAG) 2.0. Available from: http://www.w3.org/TR/ATAG20/
[7] Professions & Professionalism. 2014. Editorial Guidelines. Available from https://journals.hioa.no/index.php/pp/about/editorialPolicies#focusAndScope
[8] Michel MJ, Knouft JH. 2012. Niche variability and its consequences for species distribution modeling. PLoS ONE 7:e44932. Available from: http://dx.doi.org/10.1371/journal.pone.0044932
[9] PubMed Central. 2011. File submission specifications. Available from: http://www.ncbi.nlm.nih.gov/pmc/pub/filespec/
[10] International Digital Publishing Forum. 2010. EPUB 2.0.1. Available from: http://idpf.org/epub/201
[11] Spehar I, Kjekshus LE. 2012. Medical management in Norwegian hospitals. Professions & Professionalism 2:42–59. Available from: http://dx.doi.org/10.7577/pp.v2i1.178
[12] Piez W. 2010. Fitting the Journal Publishing 3.0 Preview Stylesheets to your needs: Capabilities and customizations. In: Journal Article Tag Suite Conference (JATS-Con) Proceedings 2010 [Internet]. Bethesda, MD: National Center for Biotechnology Information (US). Available from: http://www.ncbi.nlm.nih.gov/books/NBK47104/
[13] Piez W. 2011. Taming the beast: JATS data, non-JATS data, and XML Namespaces. In: Journal Article Tag Suite Conference (JATS-Con) Proceedings 2011 [Internet]. Bethesda, MD: National Center for Biotechnology Information (US). Available from: http://www.ncbi.nlm.nih.gov/books/NBK62086/

About the authors

Trude Eikebrokk is a senior librarian and leader of the Digital Services group in the Learning Centre and Library. She is the technical manager of the publishing platform Open Access Journals at the Oslo and Akershus University College of Applied Sciences.

Tor Arne Dahl teaches web technologies in the Department of Archivistics, Library and Information Science at the Oslo and Akershus University College of Applied Sciences. He has previously worked as a software developer and is working on a PhD in The Study of Professions concerning information architecture.

Siri Kessel is an assistant professor at the Department of Computer Science at Oslo and Akershus University College. She is one of the coordinators of the Master's Programme in Universal Design of ICT, teaches "user diversity and ICT barriers", and is doing research in this field.

Subscribe to comments: For this article | For all articles