# EPub as a way of packaging scholarly resources?

There's a lot of talk at the moment about EPub as a repository-relevant format, both for distribution, and as a packaging format.

Recently Brian Kelly has been exploring EPub delivery, and there's a really good discussion going on in the comments of his blog where people are suggesting tools and discussing the role of EPub. As for tools we already have EPub support in our content management (ICE) and repository (The Fascinator) toolkits, I'll post a summary of those features soon.

In this post I thought I'd go through some of the **thinking** we've been doing at the Australian Digital Futures Institute on how to package Scholarly documents for the web, and for reposit, so they can be:

- Treated like paper and published as PDF.
- Used as plain-old web pages.
- Deposited in repositories – reposited.
- Used in Learning Management Systems (LMSs – VLEs for the UK readers).
- Blogged.
- Linked-to at at least the section level, maybe at the paragraph level.
- Bundled with or linked to data that supports them.

This means, we have been considering how documents can be packaged so they can be adapted the **continuum of reading experiences** that I have tried to lay-out here:


*Figure 1: The continuum of reading experiences*

Different points on the continuum[1] allow for different affordances:

- **Paper** → You know what paper is.
- **Digital Paper,** by default PDF → PDF is (at its simplest and worst) just paper for putting on the computer – it can be set up for annotations, but only under some circumstances, and in my experience it usually isn't. Not ideal for reading on small screens depending on how well structured the file is but it might be fine on a pad-type device if the resolution suits the screen size and your visual acuity.

  There are new digital-paper type formats turning up now that the iPad has arrived – some of these are no more than pictures of text, far from ideal for scholarship.
- **Re-flowable but mostly static reader** apps and devices like Kindle (ePub, Mobi) → I've conflated two things here; the mode of packaging, putting a document into a re-flowable container, and the application that you use to read it. Depending on the app, the platform, the device it's running on and the distribution channels involved you may or may not be able to annotate, or highlight and share your annotations, but you should at least be able to resize the text to suit you. These are reading-specific environments, not open-ended browsers like you use for the web. Some of the hardware used is general purpose such as phones, while some is optimised for reading, eInk devices like Kindle.
- **The old-fashioned web** → There's nothing to stop 'documents' going on the web as HTML (apart from the continued lack of accessible tool-chains for ordinary authors to create long and/or complex content

1 I love continuum almost as much as I love skiing – you don't get to use a double 'i' or a double 'u' all that often.

that is). This is what we use for USQ courseware at present.

- **The new web**, where a document can be an HTML 5 app that can host data visualizations via the canvas element and adapt itself to different device-environments; here we could have very rich annotation services for local and shared use, and essentially distribute reader software with every document.

In order to support all this, I have been talking with various people about the idea of delivering documents in a sort of mega-package, which would be a zip file stuffed full of document stuff:

- The source-files (word processing, XML, LaTeX, whatever).
- HTML
- Images / video (yes – video needs is tricky – you need to transcode for different environments, how many formats you put in a package is an open question)
- Data (if appropriate) and links to data (which really should be appropriate to all modern scholarship).
- And most importantly a whole lot of manifests and tables of contents for loads of packaging formats:
  - HTML 5 manifest and scripts for a self-contained reader – such as the Paquete project we have at ADFI.
  - IMS content package manifest.
  - EPub TOC.
  - METS package manifest.
  - Atom feed/archive.
  - OAI-ORE with links to the local files.
  - Bagit.

Yes – all in one Zip file; to give the package the best possible chance of survival in the wild.

Of course once the package has made it into a repository then it could serve up just the bits that people need and/or push the content on to other services. Duncan Dickinson suggest the use of "stripper tools" to remove the unwanted parts.

I summarised this idea on that Twitter thing:

> ptsefton Working more on "Scholarly HTML" as a magic zip file that can be filtered-down to EPub for static reading but come alive in HTML 5.

Well, it turns out that the EPub e-book spec actually covers this use-case quite well. It's actually three specs:

> **".epub"** is the file extension of an XML format for reflowable digital books and publications. **".epub"** is composed of three open standards, the **Open Publication Structure** (OPS), **Open Packaging Format** (OPF) and **Open Container Format** (OCF), produced by the IDPF. **".epub"** allows publishers to produce and send a single digital publication file through distribution and offers consumers interoperability between software/hardware for unencrypted reflowable digital books and other publications.

EPub allows for more than one version of the same thing, and specifies a basic set of image formats that must be supported, while allowing for inclusion of richer content via the `object` element in HTML. Importantly, ePub readers are required to ignore scripts, but there's no reason that scripts can't be there, so it would be possible to ship, say a chemistry thesis with a built-in 3d molecule viewer which would view as on-screen paper, but come alive on your new Android A4 pad-device.

We would have to work out how to make **documents which contain declarative links to data files,** such as chemical markup language, wrapped around an place-holder image, with the smarts to ask their environment "Hey, environment, can you render CML? " and if it can, then show a 3d molecule using something like CanvasMol (which doesn't support CML, yet). Or music research could come with data files that can be rendered in-browser via something like VexFlow, or fed to a synthesiser.

This means that repositories in particular could be set up to support certain kinds of renditions for a certain

period, like having JMOL for rendering molecules for say three years, but after that time could just let the document fall-back to having a link to the data file. It could also mean that we could ship an HTML 5 app for reading and annotating a thesis which would work for reading on an old-web browser, but come alive and get all interactive in a more modern environment.

# And the point of this is?

**A user** finds something.epub on their desktop. The worst case is that they have to Google to find out what kind of thing `.epub` is, but then they should be able to do something with it, read it in a reader, put it on their phone. Sure, the ePub they're using is bloated with extra stuff that they don't necessarily need but it least it should be usable.

**In a repository context**, the repository could use the ePub or IMS tables of contents for the content and deliver an in-browser book-viewer. We have some technology for this, or there are things like bookworm (via Steven McPhillips in the comments to a previous post). The repository could also deliver the PDF, and original source document. This contrasts with the current approach where we are really just relying on convention. Upload a Zip file to Eprints and it unzips it into data streams and looks for a default.html file for example. And you have all that stuff archived, giving more chances that you have backed a winning format for the future re-use and/or re-discovery of your content.

# But does it make sense?

Am I talking sense here – bundling lots of stuff into one package?

After all, if we really got our ducks in a row we could set up systems that delivered ePub for mobiles, an OAI-ORE or METS package for repository deposit, IMS content packages for LMSs without any of this extreme package-overloading.

Comment, please.

# How to make this happen

So, if this does make sense, and that's far from a given *how would we go about doing it*?

One idea I'm considering is that we could enable a lot of this via SWORD services. It might work like this.

- Write a long document in Word and decide to deposit it.

  (Your authoring environment, could be Word itself, could be a Drupal, could be SharePoint, has a 'deposit' button.)
- Click `Deposit`.
- The authoring environment has a conversation with the servers/services it knows about.

  One service would be: 'convert this to HTML'; and other services might offer more things that you can do with the HTML: 'convert this to EPub', 'blog this'.

  Another service would be 'deposit this in a repository'.

  Another would be 'add data' – in interactive service to help locate and link data that needs to be included in or referenced in the package. This service would have to be integrated into a research data repository, and obviously the researcher would have to be inducted into its mysteries.

- You choose the services you want and click Go.
- Your authoring environment orchestrates a series of SWORD transactions.
    - First-up it would make a minimal EPub package with just the word doc and a PDF in it. The HTML component would be just a README that points to the source file and PDF.
    - Posts your document to a server that generates an HTML rendition and returns a Zip file with the original Word document plus an HTML rendition. (This service could, of course be local to your machine).
    - Posts the result to a blog service – relaying its questions back to you. “What category?” “Do you want Twitter with that?” (You don't necessarily get anything added to the package except maybe a record that it was blogged, and where, for the scholarly record.)
    - Posts the result to an ePub generator.
    - Sends the whole (now quite portly) package to your institutional repository.
    - And so on...




This post was written in OpenOffice.org, using templates and tools provided by the Integrated Content Environment project and published to WordPress using The Fascinator.