Accelerated Publishing

Tools and Processes to Publish Professionally, Efficiently, and Effectively

Sean Harrison

BOOKGENESIS PRESS
DEKALB, ILLINOIS

Accelerated Publishing

Printed in the United States of America

Contents

1

Writing Tools

As an author, you should write with the tool that you are most comfortable with. Don't let anyone tell you should or shouldn't use this or that tool. Everyone has different preferences. I have used a wide variety of tools for writing. They all have strengths and weaknesses.

Whatever tools you choose to use for your writing, however, your goal should be to *focus on content and structure, and not on formatting.* Any formatting that you apply to the manuscript should communicate the structure and meaning of what you are writing. There is no reason for you to spend any time or thought at all on what fonts you should use, or whether your headings should be bold or italic, large or small.

All of the tool-usage scenarios that we discuss, therefore, will focus on *structured writing*. The goal is to enable writers to produce manuscripts that clearly communicate the content and the structure of the writing, and which can easily be brought into a professional publishing workflow when the draft has been completed.

Microsoft Word

I have seen many people express hatred for Microsoft Word. If you don't like it, you don't need to use it. There are a variety of other viable options now.

If your editor requires you to submit a Word manuscript and you don't use Word, most word processors can export your manuscript as a Word document. Use the "Word .docx" format, which is widely supported. If you use a text editor or blogging software to do your writing, there are

ways to convert plain text formats or blogs to Word documents (see below, "Plain Text Formats and Editors" and "WordPress and Other Blogging Platforms").

However, some people do prefer to write in Microsoft Word. I am one of them, probably because I have been using it in my editorial work for almost 20 years, and I know exactly how to use it for writing and publishing. (In fact, I am using it right now to write this section.) It doesn't get in my way, it gets out of my way. It has all the features I want for writing anything, and I turn off or hide or ignore the features that don't.

If you are going to use Microsoft Word to write, learn and use the features it provides to make writing better and easier. The chapter entitled "Writing and Editing in Word" discusses in depth getting the most out of Word as a writing and editing tool.

Plain Text Formats and Editors

In the past few years, "plain text" has seen a resurgence of popularity as a writing and editing format. Formats like Markdown, which had their genesis in the world of software development, have slowly been spreading in popularity. As a result, nowadays there are a lot of good tools available to make plain text a viable professional writing and editing format.[1]

Before the computer age, authors either wrote by hand or used a typewriter. These are plain-text authoring environments. We have become so used to word processors that we forget how recent an invention they are.

When using a typewriter, an author could not italicize or bolden text; underlining was just about the only way to indicate emphasis, and it required back-spacing over the text and making a second pass using the underscore key.

When we all started using email in the 1990s, for the first few years emails were all written in plain text. So a number of conventions for indicating formatting arose: _underscores_ around a word or phrase would indicate emphasis/italics, *asterisks* would indicate bold, and so on.

Even after email moved into an era of rich-text/WYSIWYG composing, a lot of people still use these conventions in writing email and a variety of other kinds of documents. For example, Facebook comments can be quite long, but (as of this writing[2]) they have no formatting capabilities — if commenters wants to indicate bold or italic or the like, they will use

Recommended Plain-Text Editors

Name	Platforms	Description
Sublime Text (www.sublimetext.com/)	Windows, macOS, Linux	One of the most capable and high-performing text editors, easy to use and configure, and available on all major desktop platforms.
iA Writer (ia.net/writer)	iOS (iPhone, iPad), Android, macOS	Provides a clean and simple writing environment, with Markdown preview, and syncing via iCloud and Dropbox.

underscores and asterisks in these same ways.

There are now several popular plain text formats that can be used to produce rich-text output. The most popular of these formats is called Markdown.[3] Markdown has the distinction of being very easy to create and easy to read in a plain-text editor. So, you can author your manuscript using a plain-text editor, then convert it to rich text during intake to the publishing workflow that you are using.

Another benefit of using plain text is that there are excellent text editing apps available on every platform, and you can use a file syncing service like Dropbox or iCloud or OneDrive to sync your manuscript files between your computer, your tablet, and your phone. This makes it easy to work on your writing wherever you are, with whatever computer you happen to have—the one in your pocket, the one in your satchel, or the one on your desk.

I am providing a very short, very opinionated list of "Recommended Plain-Text Editors" (table, above). This list only provides one or at most two choices per platform, the focus is on quality and usability, and price is not a factor in the decision. If you are publishing professionally, then my assumption is that you can afford a good text editor.

WordPress and Other Blogging Platforms

If you want to write your manuscript online and invite others to interact with what you've written along the way, one approach is to use a blogging

system for your writing. Most of the popular blogging software providers will allow you to create a new free blog and use their platform to write your manuscript. You don't have to make your manuscript blog public if you don't want to—it can be private, and you can invite only the people you want to invite. When you are ready to publish your manuscript, you should can then export it from your blog, and a good publishing workflow should be able to import it directly as a manuscript, retaining all of your formatting and so on. One thing to keep in mind, of course, is that blogging platforms are designed for creating webpages, so they don't really have any support for things like footnotes and endnotes. They are therefore best used for simple writing, such as fiction, that doesn't contain those features.

2

Writing and Editing in Word

The majority of authors and editors have traditionally used Microsoft Word, and for good reason: It is very, very good as a writing and editing tool.

Some people have poo-pooed Word, and word processors in general, as not supporting structured writing and editing. It is true that most authors (and many editors) don't use Word as a structured document editor. But Word has very rich structural capabilities. Once you learn to use these capabilities, it is an excellent writing and editing tool. In my own work I and the teams I have worked with have used Word very effectively to supply hundreds of manuscripts, from very simple to very complex, to the page typesetting and ebook composition processes.

In this chapter, I will explain how to use Word to do structured writing and editing. In a later chapter, I will talk about how to hook Word into an professional publishing workflow as a content source.

Paragraph Styles

Many authors are unaware of Word's built-in support for paragraph styles, but paragraph styles are at the foundation of a professional publishing workflow, and learning to use them effectively can make your life as an author or editor much less frustrating.

For example, rather than using ALL CAPS or bold or italic type to indicate a first-, second-, or third-level heading, use one of the heading styles: "Heading 1," "Heading 2," and "Heading 3." These styles are readily available both in the styles pane and on the keyboard.[1] Similarly, use "Body

Text" for regular paragraphs, "List" for a list, and so on. Word comes with a pretty good selection of styles built in. You can use these styles rather than extra returns, tabs, and bold/italic font formatting to indicate what each paragraph *is*, not just how it *appears*.

The built-in selection can, however, be improved. For example, "Chapter Title" is a better style name for the chapter title than "Heading 1". So I have created a Word Manuscript template that adds a number of useful styles and keyboard shortcuts.[2]

Character Styles

Just as paragraph styles are better than using Return and Tab to format paragraphs, character styles are better than using direct bold or italic to format text. The primary reason character styles are better is that they give designers much greater flexibility in how they display bold and italic text. With direct bold formatting, the bold text will always be displayed with the bold weight of the same typeface that was used for the Roman (regular) weight text, and the italic version of the same typeface will be used for italic text. But sometimes this is not what designers want: They might want to use a semi-bold weight of the same typeface for bold, or even a different typeface altogether, to represent bold type. Similarly, they might want to use a different typeface for italic type. When you use character styles rather than direct formatting, you make it possible for the designers to have this flexibility without requiring them to hunt through your manuscript for instances of italic and bold text. They just need to define the style for the italic or bold character style, and they will get the results they wish.

For this reason, the BlackEarth Word Manuscript template binds new character styles called "bold" and "italic" to the keyboard shortcuts that are usually used for direct bold and italic formatting: `Ctrl+B` and `Ctrl+I` on Windows, `Command+B` and `Command+I` on macOS. If you write your manuscript from the beginning using this template and these common shortcuts, all of your bold and italic text will already be styled correctly for production, without your needing to do anything additional.

There are also situations where a different typeface is needed in-context to accommodate a different language or script. For example, a lot of academic manuscripts in Biblical Studies and Theology use Greek and

Hebrew right in line with other text. In those cases, rather than limiting the design only to typefaces that support Greek and Hebrew, we can use "greek" and "hebrew" character styles. Then the designers can choose whatever typeface they wish for headings and body text, and use different typefaces for Greek and Hebrew type if they so choose.[3]

Footnotes and Endnotes

Word's footnote and endnote capabilities are one of the things that makes it capable as a professional publishing tool. Generally speaking, you can use Word's footnote and endnote capabilities while writing and editing your manuscript. The footnotes and endnotes *should* be carried through the entire publishing process, with correct automated numbering and bi-directional linking in digital formats.[4]

Hyperlinks

{TODO}

Bookmarks

{TODO}

Tables of Contents

{TODO}

Indexing

{TODO}

Tracked Changes (Redlining)

Word's change tracking or "redlining" functionality is excellent. To use it, just turn "Track Changes" on in the Review ribbon, and Word will automatically track all changes that you and others make to the manuscript.

Comments

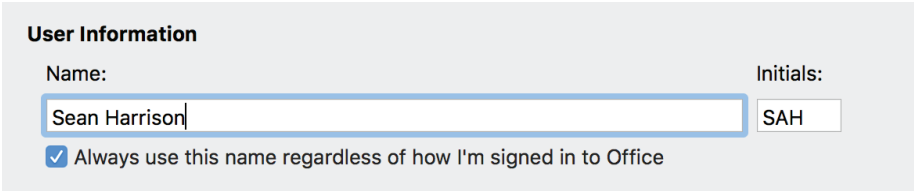
Word's built-in comment function enables various people to read a manuscript and make comments on it without changing the text. "Insert > Comment" is available on the "Insert" ribbon or menu. Others can then reply to the comments or mark them as resolved. This makes a very natu-

ral and easy-to-use way to interact regarding the content of a manuscript.

User Information

Tracked changes and comments are connected to each user, and you can hover the mouse over a change to see who made it. However, on a lot of teams, the user name is often something like "Software Administrator," because the actual user didn't set up their own copy of Word. You want the correct name of the user to appear so that, when collaborating with others, you can have a conversation if one of you disagrees with a change. (Your user name also appears whenever you insert a comment.)

So, to make sure the correct user information appears, open Word's "User Information" preferences and type your name and initials into the provided fields, if they are not already there.



Document Properties and Variables

{TODO}

Capabilities of Word to Avoid Using in Manuscript Writing and Editing

Direct Formatting of Font, Paragraph, Column, Section, and Page

{TODO}

Pictures, Clip Art, Shapes, Charts, Text Boxes, WordArt

{TODO}

Questions People Ask about Word and Word Documents

What about Future Proofing? Shouldn't we be archiving our manuscripts in a future-proof format?

Yes, you should, which is why you should always store your Word documents as .docx files. This is the default in recent versions of Word, but to be sure, go into the Word "Save" Preferences and look for a line that says "Save Word files as:", following which the drop-down menu should say: "Word Document (.docx)". If it doesn't say that, change it to that.

The reason for using .docx format for Word Documents is that .docx uses XML for all of the content data, and ZIP to compress all of the XML into a single file. Both ZIP and XML are industry-standard technologies—they are being used by just about everyone for just about everything. For example, an EPUB ebook is just a ZIP file with XML inside of it.

In the 1980s and 1990s, there were no standards for file formats, so software vendors created a lot of incompatible proprietary formats. But then, as the web started taking off, we began to see the benefit of being able to share content in standardized formats. There was also a strong movement toward using XML as a standard way to format content. Eventually someone came up with the idea of packaging XML inside of a ZIP file as a way of creating a cross-platform file format (ZIP is a public-domain archive compression format that was first released in 1989 and began to be very widely used to compress and archive files and folders.[5] The idea of using XML inside ZIP caught on. Since 2007, Microsoft Office[6] and Libre Office[7] have both used this method as the default file format storage mechanism. EPUB, also released in 2007, is essentially a website inside a ZIP archive, with a couple of other XML files to indicate reading order and such.[8]

Some people worry that these formats are less future-proof than plain text, but it is in fact extremely unlikely that either ZIP for compression or XML for content data will ever be superseded in the foreseeable future. The global adoption of these technologies was a process that took 40 years to complete, and it is likely that the world will continue to use them at least for the next 40 years. Even if something comes along to replace ZIP and XML as the primary storage format, the tools required to work with them

are freely available and ubiquitous, so it will always be possible to read, process, and write files in these formats.

The result of all of this is that, at least since 2007, if you are using the default .docx file format of Microsoft Word, you are already storing your manuscripts in a future-proof format, and you don't need to worry about it at all.

3

Typesetting with InDesign

Styles, Not Overrides

It is probably stating the obvious to say that typesetting should use paragraph and character styles to format books. After all, everyone uses styles, right? Well, yes, except for when they don't. The reality is that there are many situations in which what InDesign calls "overrides"—or, local formatting—is tempting. It's so easy just to apply some adjustments to the formatting and be done with it.

Dynamic Endnotes

InDesign has no built-in support for endnotes. This can be explained in various ways, beginning with its roots in magazine publishing, where endnotes don't really appear. But that explanation wears thin now that InDesign is 17 years old—almost an adult!—and has become the primary page layout system for the entire publishing industry. But, still, no endnotes. So another explanation is needed. Such as complexity? That seems hard to believe, since Microsoft Word and every other word processor has supported endnotes for 30 years. But, no endnotes. So we're left with laziness, incompetence, tone-deafness to the needs of book publishers (who strongly favor endnotes over footnotes), and/or uncaring.

Be that as it may, we need to support endnotes in our books. It turns out that there is a relatively easy way to do it, even with InDesign's built-in Word import.

Procedure for Creating Dynamic Endnotes in InDesign from a Placed Word Document

1. Place the Word document, making sure that the "Include: Endnotes"

option is checked and that all styles and formatting are preserved. This will result in the endnotes being placed at the end of the story for this document, with the endnotes styled with the "Endnote text" paragraph style, and the endnote reference numbers in the text being styled with the "Endnote reference" character style.

2. Delete the endnote reference numbers from the beginnings of the endnote paragraphs, and the double-spaces from between the endnote paragraphs.
3. Open the "Style Options" for the "Endnote text" paragraph style, go to "Bullets and Numbering." List Type: Numbered. Format: choose the kind of numbering you want: Arabic, Roman, etc. Number: match what you want—probably **^#^t**. Character Style: Endnote reference. Mode: Continue from Previous. Now all of your endnotes will be auto-numbered.

   If you have several chapters of Endnotes, all numbered from 1 within each chapter, you will need to add a new style: "Endnote First". Set all of the numbering options as above, but for the "Mode:" choose "Start At" with 1. Then apply this style to the first endnote in every chapter.
4. Search through the document for all endnote references. In the search dialog, the search text should be blank, while for the "Find Format" you should choose the character style "Endnote reference." With each endnote reference selected, open the cross-references palette and select "Insert Cross-Reference". Link to: Paragraph. Destination: Endnote text, and select the endnote that is your destination. Cross-Reference Format: Paragraph Number

When you have gone through this process, all of the endnotes will be automatically numbered both in the text and in the back matter. If the author or editor later adds a note, all the numbers will reflow without great pain. Also, if you create an interactive digital PDF, the endnote references in the body will become links to the endnotes (unfortunately, the reverse is not true). And in the ebook export process that we have developed, these links are bidirectional.

It's not a perfect process—certainly not as easy as creating the endnotes in Word in the first place—but it gets the job done.[1]

Automated Tables of Contents with Bi-Directional Linking

4

Digital Pre-Production with InDesign

A book that has been typeset in InDesign can be prepared for digital production right in InDesign. The advantage of doing digital pre-production in InDesign is that the InDesign publication itself becomes the archival form of the product, with all outputs being produced from it. It will work even for the project that has to be split up into multiple publications and combined using an InDesign book (.indb) file, such as can be done with large reference works like study Bibles. Rather than having to maintain two different and separate versions of the product, the content and production team can maintain just one, archival version, the InDesign publication, from which all outputs are created. When there are corrections, they can be made in one place, the InDesign publication, and all outputs can be updated from there. This reduces both headaches and errors and stress from the publishing process—all of which are important for the busy production team.

This chapter covers most of the techniques you need to prepare InDesign publications files for digital production. It doesn't cover the digital production process itself—that is covered later in the chapter entitled "Ebook Production."

Bookmarks: Document Structure and Sequence

Every digital publication has a digital table of contents, which is separate from and not necessarily the same as the table of contents that is shown in

the interior: It might either be more comprehensive or more abbreviated, for example.

It turns out that InDesign's bookmarks functionality provides precisely what we need in order to build the digital TOC for all digital book formats. Notably:

- The bookmarks can be nested, providing a multi-level digital table of contents for complex books.
- InDesign's interactive PDF export uses the bookmarks as the digital TOC for the PDF (it "does the right thing").
- The bookmarks can be used to build the "spine" and TOC in an EPUB production workflow.

So the process of building the digital TOC (and spine, for EPUB production) can be as simple as creating the needing bookmarks in InDesign. Then these will be picked up during the production process.

Creating the bookmarks themselves is a simple process:

1. Open the Bookmarks palette.
2. Place the cursor where you want a bookmark link to target (the location to which the bookmark will take you). If there is a heading that should be the text of the bookmark, select that heading.
3. In the Bookmarks palette, select "New Bookmark" (or use the keyboard shortcut that is probably listed there). It will add a new bookmark with the name of the selected heading, or with the name that you type.
4. You can also edit the text of the created bookmark by selecting and then clicking on the name of the bookmark (click twice slowly).
5. You can drag and drop the bookmarks to reorganize them. If you drop a bookmark (or group of bookmarks) onto another one, it will nest that bookmark inside the one it was dropped on.

That's it. Just do that for every heading in the interior that you want to appear in the digital TOC. Easy! But very repetitive. Just the kind of thing that a script would be good for.

Conditions: Print-Specific And Digital-Specific Content

In most books, some of the text is print-specific, while some other is digi-

tal-specific. InDesign's conditional text functionality provides a very good way to have both kinds of text in the product. All you need to do is set up a condition named for each different kind of text, and apply that condition where it is used. Here is how to set up a basic set of conditions:

1. Open the "Conditional Text" palette (in the Window menu under "Type and Text"). (Everything in InDesign starts with opening a palette, it seems.)
2. Select "New Condition". Name: "Print". Indicator Color: pick one that you will consistently use in all your products for the "Print" condition.
3. Rinse and repeat for "Digital" and any others that you might want.

That will give us enough to go on.

The important thing, now, is to know what kinds of things to apply Print and Digital conditions to. Here are some things that we usually do:

- On the Copyright page, we usually apply the Print condition to the print lines (those that say, e.g., "Printed in Italy" and those that have the numbers indicating which printing it is).
- We also usually include on the Copyright page a "build line" with the Digital condition, which says "Build date: <Timestamp>", where timestamp is the output date of the file (use the menus: Type > Text Variables > Insert Variable > Output Date).[1] The build line does for digital products what the print lines do for print products: They indicate to customers when the file that they have was created. This can be very useful for fielding customer service calls.
- We usually apply the Print condition to all the text on the Title Page and link an image of the title page in its place (see "Images and Frames: Including Additional Content," p. 22).
- In the interior of the produce, there might be cross-references/links to other sections, such as, "see Title of Section, page NNN." We will usually put a print condition on the page number portion of this and any punctuation that goes with it (such as the comma). We'll also make sure that the Title of Section is a cross-reference or at least a hyperlink to the given section so that it is clickable in digital products.

There may be other places you'll want to use Print and Digital conditional text in your products, but those are some of the most common.

Then, when creating print output, make sure that the non-print conditions are not visible by unselecting the "eye" icon in the Conditional Text palette. (We will discuss digital outputs at a later point.)

Images and Frames: Including Additional Content

Many, if not most, books are more than just formatted text—they also include images, text boxes, and other kinds of content that are alongside the main text. At the very least, a novel will have a title page, and for digital products this title page is probably best represented as an image. It might also have a map of the fictional world, and there might be one or more cross-promotional pages at the back. All of these are best handled as images.

There are, by my count, four kinds of images and frames that need to be included in a digital product. All three of these image types can be included in the InDesign text so that they are also included in the digital output. The three types of images are:

1. Inline images that are anchored in the interior text
2. Images that are alongside the interior text
3. Content frames that are alongside the the main text
4. Textual elements that should be converted to images for digital publication

We will discuss each of these kinds of extra content in order.[2]

1. Inline images that are anchored in the interior text

Some of the images in a book are already anchored into the main interior text. For example, a figure might be included inline immediately following the paragraph that introduces it and immediately before the paragraph that explains it. Mathematical equations, chemical diagrams, and excerpts of music are examples. Or the text might contain small images within a paragraph.

In any event, if an image is anchored in the main interior text, you shouldn't need to do anything further in order to prepare it for digital production: InDesign's output will already include everything that should

be needed for the digital production process.

That was easy. The others require a little more attention.

2. Images that are alongside the interior text

Other images in a book are not in the main flow of text. If they are anchored in the text, then you're done, as above. If they're not anchored in the text, there is one more thing to do: Create an InDesign Note that adds an image tag into the main text.[3]

Why an InDesign Note? Because that is really the only good place that InDesign gives us to create additional "tags" within a text in order to accommodate the specific needs of digital publishing.

The Note will contain the information needed to include the image in (non-PDF) digital outputs, without impinging at all on the layout of the print product. Here's what you do:

1. Place the cursor in the text at the location where you want the image to appear in digital outputs.
2. Open the Notes palette (Window > Editorial > Notes) and select "New Note".
3. Type the following into the note you just created:

   ```
   <img src="relative/path/to/image.pdf"/>
   ```

   The source (`src`) of the image (`img`) is the file located at `relative/path/to/image.pdf`. For example, if you are keeping your images in a "Links" subfolder next to the InDesign document, and the image filename is "SmileyFace.pdf", then the source will be "Links/SmileyFace.pdf".

Notes create a little red hourglass shape in the text when InDesign is in "Normal" Screen Mode. They have zero width, and they have no effect at all on the flow of text (unless, perhaps, they are inserted in the middle of a word, which is not really a good idea anyway).

The image Note will be exported along with the content and can be used by a digital publishing workflow to include the image at that point in the flow of the text.

3. Content frames that are alongside the the main text

Sometimes there are content frames that occur alongside the main text

in print, but need to be included in the main text in digital products. For example, a table might be set so that it as at the top of a page, in its own text frame with the main text flowing around it. Or a callout might be set in its own text frame between two columns in print, but needs to be inline in the ebook. In such cases, we need to do two things: First, we need tag the content in the text frame to identify it. Second, we need to indicate where to include the content in the main text. For both of these purposes we will use InDesign Notes (for the same reason we mentioned under "Images that are alongside the interior text": It's the best way InDesign gives us to provide these hints to the digital production process). Here are the steps:

1. Place the cursor in the text frame that is outside the main flow of text, at the beginning of the content. Open the Notes palette, create a New Note, and type the following into the Note:

   ```
   <pub:section-start id="a-unique-id-for-the-section"/>
   ```

   The unique section id should be something that you can guarantee will be unique in the entire product, it should start with a letter, and only use letters, numbers, hyphens, underscores, and periods. I recommend using something related to the content of the section: For example, if the section is Table 3.5, a good section id would be `"Table.3.5"`.
2. Place the cursor in the main text at the location where the text frame should be included in digital products. Create a New Note, and type the following into the Note:

   ```
   <pub:include id="the-same-id-that-you-created-for-the-section"/>
   ```

   For example, if the `<pub:section-start/>` tag was given the id `"Table.3.5",` then so should the `<pub:include/>` tag.[4]

Having tags both in the main text and in the frame alongside it makes the connection that is needed to include the frame content in the digital product.

Doing this is only needed if the sidebar content should be included into the main text at a non-heading break. If, on the other hand, the sidebar content has a heading and is included in the digital TOC bookmarks, none of this is needed: It will be included as indicated by the position of its bookmark.

4. Textual elements that should be converted to images for digital publication

The last scenario that we need to address is textual content that should be converted to images for inclusion in the digital product. In general, this is true for any combination of text and images in InDesign, and for textual elements that cannot reliably be represented as given in ebooks. A couple of common types come to mind:

- Chapter numbers and dingbat characters that are set in a special font, the appearance of which is important to the "identity" of the product.
- Artwork that combines text and images on the InDesign page (rather than, say, via placing a PDF). The most common example of this is the title page.
- Foreign language text that digital platforms won't reliably support.
- Complex tables that cannot be reflowed as lists and won't display well as tables.

For all such situations, the solution is the same:

1. Create an image file representing the visual content and store it in a subfolder that is next to the InDesign publication (usually "Links" or "Images"). The best way to create an image file is to create a print PDF of the interior, extract the page that contains the visual content, and crop the PDF page to the bounding box of the visual content. Save this PDF as the image file.
2. Insert an `<img/>` tag Note that references the image file.
3. Put a "Print" condition on any text in the content that should not be displayed in the digital product but would be displayed without the Print condition (see "Conditions: Print-Specific And Digital-Specific Content," p. 20).

Conclusion

Using the techniques outlined in this chapter, most books that have been typeset in InDesign can be prepared in this way in somewhere between an hour and a day, depending on the complexity of the book and the skill of the specialist. Long and complex projects, of course, can take much longer.

5

Archiving Content and Projects

Why You Need to Archive Your Content

The traditional publishing workflow can be seen as a linear process in which content moves from one station to another; the content changes hands in a "hand off," which might be the only time that the two roles talk with each other about the project. (If the hands that are doing the handoffs feel particularly disconnected, they might talk about it as "throwing the manuscript over the wall.") In that kind of workflow, the final form of the content before being printed is usually the typeset pages, and the archive of the typeset interior is the *de facto* content archive for the project.

Most ebook production has been based on converting typeset pages to distributable ebooks. This is usually a one-time process that many publishers outsource to the lowest bidder. If there are changes in the content, often these changes don't get into all editions unless the publisher invests in a manual reprint correction process that covers all bases. So there are often several versions of a successful product, and no one is sure which one is the "right" one. Worse, opportunities to repurpose the product and distribute it in a variety of forms are missed, because the product content is "locked up" in print formats.

We can do a lot better than that; in fact, we *need* to do better than that, if we are going to publish our content to all of the formats and platforms that are available and still maintain a sense of control over what is going on with each. For anyone who has worked in a busy publishing operation with hundreds of products and many separate editions of the same con-

tent sets, it is clear that there has to be a better way.

What if a publisher poured all of their content into a central archive that was indexed and linked, so that all of the content was always available for any purpose that the publisher could imagine, with the assurance that the archived version is the corrected, up-to-date edition of the content? What if conversion to ebook and other new formats was a simple, automatic or mostly automatic process from this archive? What if content changes and corrections could be propagated automatically to all of the places where the content has been published? What if a publisher could make all of their published content available on a new platform simply by defining a single transformation process?

In this chapter I will be describing a set of practices that will enable a publisher to curate all of its content in a central archive, from which every published form of it can be dynamically drawn.

Server-Based Versioned Archiving

The central content archive is the hub of an XML publishing workflow. All content flows into this hub for permanent curation, and all content flows out of it for publication.

Subversion: The Recommended Content Archive Platform

Using Subversion to house content archives on web servers has several advantages:

- The "centralized" content and authoring model of Subversion fits the production workflow better than the "distributed" model of git. Git's model was designed for distributed open-source software projects and is a good fit for individual projects (see "Git: A Good Option for Individual Projects," p. 30).
- All content changes are automatically versioned, and each version is accompanied by the username of the person who made the change and a descriptive log message.[1] This makes it easy to compare any two versions of the content at any later time, and to see what happened during the course of product development. It is sometimes very valuable or even crucial to be able to do this easily, but it cannot

Choosing a Version-Control System

	Subversion	git
paradigm	centralized	distributed
implied workflow	centralized	non-linear, non-centralized
commit	one-step, remote	two-step: local commit, remote "push"
working copy	any path within repository	whole repository
implied repository organization	monolithic: one repository	atomistic: many repositories per server
client cmd	svn	git
programmable hooks	yes	possible, with difficulty
extension languages	several bindings	several bindings
WebDAV	yes	possible, with difficulty

be known in advance when it will be needed. When versioning and log messaging are just part of the daily process, it becomes trivially easy to find such information later as needed.

- The archive, or a small part of it, can be "checked out" to a local working copy, which makes a very convenient and easy-to-learn way of working with the contents of the archive. On our teams, the editors have all gotten proficient at working with Subversion in this way.
- The archive is available as a WebDAV shared filesystem. This means that non-technical users can mount the remote archive to their computer filesystem and edit files as though they were local. In some cases, this can be the most effective way for users to interact with the content. (I once managed the editorial development of a large, multi-year, multi-million dollar project in which the CEO of the publishing company wanted to read and edit the entire manuscript late in the editing process. Rather than saddling myself or someone else with the chore of emailing files around, I set him up with a network share to the manuscript. At any time, he could open the latest committed version in the archive and edited it "in place," saving his changes as versioned commits but having the exact same user experience as opening a file from the local network.)

- All XML content can be validated when it is "committed" (saved) to the archive, ensuring that the content matches the publisher's requirements for that kind of content. See below for more about this important part of the system.
- All content can be automatically indexed, and published to the web or other places, every time a change is committed to the archive.

Some of these advantages can be achieved by using any server-based version control system. But Subversion is the only platform that allows us to combine all of these advantages at once.

The process of setting up a Subversion server has been well-documented elsewhere; it requires a reasonably-capable server instance running Apache on any of the major server platforms (Windows, Linux/UNIX, OSX).[2]

One thing worth pointing out with this kind of arrangement is that there is no need to spend any time at all developing a custom content authoring and editing environment. A lot of developers are spending a lot of time and energy doing that, but it's not necessary. Instead, we can work with our content using normal files on the filesystem, as we have always done, and make use of the full range of software tools that have been developed to work with files in this way. The difference is that these files are backed up to a server and are available to everyone on the team, just as they would be if we had invested in a web-based environment. In other words, we get the benefits of both web-based tools and filesystem-based tools, without the limitations of either. The downside? A thin layer of skill that can be learned by most professionals in a few hours of training.

Git: A Good Option for Individual Projects

Git is a distributed version control system, which means that every working copy is a full copy of the archive. Distributed version control has become very popular among software developers, because it enables each developer to have a full copy of the repository, for the team to work in a less centralized way. Git's distributed nature is not ideal for building a publisher's centralized archive platform — it lacks many of the advantages listed above for using Subversion for this purpose. However, it is very well-suited to individual projects.

If you have a writing project that is in fact distributed among several contributors, or if it is a project that stands alone, you can always begin the project using git, and then bring it into Subversion for production archiving later. For example, I have begun this book project by creating a public git repository on GitHub.[3] This enables other people to contribute sections and then submit them via git's built-in workflow. It is an appropriate model for an open-source book project like this one. Later, I will probably publish it through an automated workflow connected to a Subversion-based book archive. When that happens, I will bring the book into the Subversion archive.

The nice thing about this approach is that both git and Subversion can be used for a project at the same time. For example, the content development can continue being done via git, and these changes can periodically be pulled into the Subversion archive and incorporated into the production workflow. (Explaining exactly how to do this is beyond the scope of this chapter. A later chapter?) {TODO}

Archive Layout

We recommend that every publisher set up a single Subversion repository that contains all publisher content. Depending on the size of your publishing operation, this single repository might grow to hundreds of gigabytes over the years; there are no size limitations, and Subversion is remarkably efficient in the storage of content (the server-side archive is often smaller than a checked-out working copy of the same archive, even if the archive contains many separate versions of the same content, because Subversion is so efficient in compressing files and content revisions, even of binary files).

Here are the folders that we recommend as a starting place:

https://my.archiveserver.com

/products/ — all of the content for each product is inside this folder.

It usually makes sense to put each product in its own subfolder within the products folder.

Product-One/ — all the files for the first product.

manuscript/ — the product manuscript.

interior/ — the typeset print interior.

cover/ — files for the print and digital cover.

outputs/ — outputs for distribution: interior PDF, digital PDF, ebook formats, app data packages, files for the website, etc.
stylesheets/ — stylesheets specific to this product.
Product-Two/
…
/stylesheets — global cascading stylesheets for the entire product line
/authors/ — information about each author can be put in this folder.
/editors/ — each of the editors can have their own subfolder, in which they can store their manuscripts-in-process and archive them when finished.

The reason we recommend this kind of folder arrangement as a starting point is that it has proven to be a good way to organize everything:

- People who are involved in product development for a particular product can access that product folder without needing to check out a copy of the entire archive; this proves important as the archive grows in size and can no longer fit in the working file space of some users' small hard drives. For example, if the publisher has 1,000 backlist products, but each team is working on 3 products at a time, the members of a team can check out working copies of the three products they are working on, and collaborate on those files in almost-real-time, without taking up hard drive space with the rest of the archive (which might be hundreds of gigabytes by that point).
- You can use access control to limit access by product and by output build or build type to particular users or groups. (For example, a particular app developer might get their own output builds that they have access to, but they shouldn't access other developers' builds or the product sources.)
- Shared stylesheets have a folder location (/products/stylesheets) that is named in a way that is analogous to the product-specific stylesheets (/products/Product-Title/stylesheets). This enables the creation of a system that collates a cascade of stylesheets from the global and product-specific folders.
- The same approach can be used for other resources, such as fonts,

shared and product-specific media, and so on.

Interfacing with the Archive

Once you have a Subversion content archive set up on a server, there are several ways to view and edit the content. You can use a Subversion client and check out working copies, or you can mount an archive directory to your filesystem and use it as though it were a local filesystem. Either way, your content files are being backed up on a server every time they are committed, a copy of every version is being saved, and you are using familiar desktop software to work with the files.

For casual users (such as authors or reviewers) who have a much lower level of investment in the "system," it might be worthwhile to provide a web-based way for them to interact with the content.

Check out a local working copy of (part of) the archive

The way that we most often work with the content archive is that individual contributors will check out a working copy of one or more product folders to their own computer system. This requires having a Subversion "client"—a program that interfaces with the server—installed on your working machine. There are several good choices here (see below). Once you have a Subversion client installed, the typical workflow is as follows:

1. **Check out a working copy of some part of the archive.** You'll put your working copy into its own subfolder. So, for instance, you might check out https://my.archiveserver.com/products/MyBook to a local folder called MyBook, and inside this folder the files will be a copy of what is on the server.
2. **Create new files and "add" them; edit or delete existing files.** You can create and edit files as normal. To add a new file to the repository, you have to execute the Subversion "add" command on that file. Similarly, to delete a file from the repository, you have to execute the Subversion "delete" command on that file.
3. **"Update" your working copy with changes that others have made.** If more than one person is working on a set of files, then you will need to update your local working copy to make sure that you have others' changes before doing your own work.

4. **"Lock" files or otherwise communicate about who is making changes to which files.** Subversion includes a locking mechanism, by which users can communicate, "I have this file and am working on it." This is useful to avoid conflicts (incompatible changes by different users) when a number of people are working on the same set of files, and so many changes are being made that it's not convenient to communicate about it all the time. In practice, it is often easier simply to communicate with each other about who has which files. (When I was studying to become a private pilot, the instructor said it was always important to know who was in control of the airplane at any time, so we used a hand-off sequence: one would say "You've got the controls," then the other would confirm with "I've got the controls." A similar sequence might be overkill in discussing files; but email makes it very easy to communicate with team members about who has control of a particular set of files at any given time.)
5. **"Commit" changes to the repository.** Until this point, all changes are "local" on the user's own system. To get these changes to the archive server, they have to be "committed," at which point the changes are transmitted and a new revision is stored that contains all of the changes in that "commit."

Subversion Clients

All Platforms: svn

If you are comfortable working with the command line, the svn command is standard across all platforms. It is pretty easy to use (as command-line apps go), and it's the same everywhere.

Most users, however, haven't seen a command line since, say, 1991 and will want to use a desktop client for their day-to-day work. Non-technical publishing professionals (editors, designers) are much happier when they can work in a familiar interface.

Windows: TortoiseSVN

We have a lot of users who are using Windows; the best client by far on Windows is TortoiseSVN. Tortoise is a Windows Explorer shell extension that adds a bunch of entries to the right-click Explorer context menu, and

the choices that are shown are those that are appropriate to the context. Because Tortoise is an Explorer extension, archive actions can be done from within any Explorer window, including a File SaveAs or Open dialog inside an application. This flexibility makes it very easy to integrate archive actions into a Windows workflow.

Unlike TortoiseSVN, every other Subversion client is a standalone application that you work within instead of using the system's own Explorer/Finder program.

Mac OS X: Cornerstone

We have been very happy with Cornerstone on Mac OSX. It is very stable, has good keyboard shortcuts, and easy to learn.

Linux GUI: SmartSVN

SmartSVN seems to be the best current GUI tool on the Linux desktop.

Every Platform: Your Web Browser

A web browser isn't a client *per se,* because you can't edit a repository this way, but at least you can browse the content and see what is there.

Mount the Archive as a Web Folder

Another option that is especially attractive to the most non-technical users is to mount the Subversion archive as a "web folder." This is possible because Subversion, when hosted by Apache, uses the WebDAV protocol. On every major desktop platform there are tools available to enable Subversion archives to be mounted to the filesystem.[4]

The advantage of working this way is that, once the archive folder or subfolder is mounted to the system, the user doesn't need to know anything about working with Subversion; they just work with the files normally as though they were on the local filesystem. For example, we had an executive editor, the CEO of a large publishing company, review and edit all the content for a large complex project through this method. As far as he was concerned, he was opening and editing files in Microsoft Word, and the entire version control infrastructure was transparent to him.

That said, there are a couple of disadvantages to working this way:

- Depending on how fast the user's internet connection is, they might experience a real lag in file operations that are actually taking place over the network.
- There is generally no way for the user to create a custom log message to describe what the commit contains — when they hit "Save", a new commit is created.

Windows and macOS: WebDrive

Although macOS has, and Windows used to have, WebDAV support built in, it is much easier and more reliable to install a third-party client called WebDrive (https://southrivertech.com/products/webdrive/). Once it is installed, you can "mount" the archive address on your system and use it as though it were a local hard drive.

Linux and UNIX: davfs2

Install davfs2.[5] Then use `mount -t davfs` to mount the archive server to a local path. (You can also add an entry to the `/etc/fstab` file in order to automate this process; read the project documentation for details.)

Web-Based Archive Interface

It is a fairly straightforward exercise for a good web programmer to create a web interface that will browse, display, and even edit the contents of a content archive. Such an interface could be constructed using the following components:

- Mount the archive to the web server filesystem using one of the tools available for that purpose (WebDrive, davfs, etc.).
- Create a web application than responds to requests for paths by examining that filesystem and serving the content that is there.
- For every XML content schema that is supported, the publisher provides an XSLT stylesheet that the web application uses to convert XML content to XHTML on the fly and serve it to the viewer.
- CSS stylesheets, both global and product-specific, are also linked so as to provide a well-designed view of the content.
- A "comment" button on each page can make it possible for reviewers to post comments about the content in each file.

- Optionally, an "edit" link can provide the raw XML in a simple <textarea>, allowing copyeditors to make small changes to the content without having to get a working copy of the product.
- More ambitiously, a "WYSIWYG" editing widget, such as TinyMCE, can be used, with the caveat that such tools can only work with content that follows the tool's own interpretation of HTML. Any content that is more technical and precise is liable to be messed up by such tools — let the buyer beware!
- If editing capabilities are going to be provided, then the web developer will want to use the Subversion lock functionality to lock files that are being edited.

That's the basic shape of what such a system would look like. In our work, we have been satisfied with a system that allows users to view and comment on content files, but not to edit them in the browser — at this point, we haven't felt that it was worth the investment of time and effort to provide a browser-based editing environment. If we did have a situation like that, then it would make sense to provide browser-based editing.

For example, if I were again starting a large writing and editing project that involved dozens of authors, editors, and reviewers, then I would most definitely take the time to set up browser-based editing of manuscripts, because it would save me from the endless chore of emailing manuscripts to everyone all the time. But I probably would make sure that the content requirements stayed well within the confines of "normal" HTML during this stage of the project, and only apply technical coding to the files after the authors and content reviewers had done their work; and once the technical coding had been applied to the files, I would "turn off" the WYSIWYG editing component for that content, because I would want to avoid any opportunity for the markup to be ruined by a tool that is too clever by half.

There are many specialized "content management systems" available. Most of them require working within a custom interface, usually in a web browser. By comparison, the approach we are outlining here is superior in almost every way for working professionals who are interacting with

content every day.

Archive Automation and Content Auditing

Subversion allows other processes to be "hooked into" the process of committing files to the archive. These "hooks" can occur at several stages of the commit process, but we will focus on the two most common: pre-commit auditing, and post-commit automation.

Pre-Commit: Content Auditing

A pre-commit hook can be used to verify the format and integrity of the files that are being committed to the archive. This process is called "validation" or "auditing." ("Validation" is a technical term for verifying the format of XML files; "auditing" is a more general term.) If the content that is being committed does not meet the standards that have been established, the commit can be blocked.

This is a very powerful mechanism: You can ensure that all content that is committed to the archive does in fact meet the standards that you have established for the format and integrity of the content. You only have to program the pre-commit hook to do the auditing/validation that you want, and to return with a non-zero exit status if the committed files do not pass the audit. It is also beneficial to provide a meaningful error message to the user who attempted the commit, so that they can correct the error and commit successfully next time.

For example: Suppose that editors are working in Microsoft Word, but they should only use certain features of Word (paragraph and character styles, tables) and not others (text boxes, bold and italic font formatting). The archive server can be programmed to audit each Word document that is committed, and only allow the commit to succeed if the documents use the allowed features.

Post-Commit: Process Automation and Notifications

The archive server can also have a post-commit hook, which is a process that runs after the commit completes. A wide variety of tasks can be automated at this point. Three of the most common are automatic file conversion, indexing, and notifications.

Automatic File Conversion. Going back to our previous example, ev-

ery time an editor commits a Word document to the archive, once the commit has succeeded, the server can automatically convert the document to HTML for use on the web and in ebooks, or to ICML for use in typesetting with InDesign.

Automatic Content Indexing. Similarly, the content in the committed Word document can automatically be added to a full-text search index, so that users can find content related to a particular search term in real time as the content is being developed. Or, this step can be saved until the content is finalized and publish-ready.

Automatic Notifications. One of the most common uses for post-commit hooks is to notify various people when particular files are committed, depending on the rules that you set up. For example, an editor or project manager can be notified whenever files are committed in a project that they are overseeing. Or suppose you license content to app developers, and you keep each developer's outputs in their own subfolder. The server could automatically notify the app developer whenever files in their outputs are updated, so that they can update their app.

Appendix A: About Publishing XML

Publishing XML defines a shared set of XML vocabularies ("schemas") for professional publishing. The purpose is to define an XML flavor that covers all publishing needs while maintaining maximum compatibility with existing systems. It fills a gap in the publishing ecosystem: There has been no comprehensive set of schemas, and corresponding tools, to cover the needs of publishing in general.

Publishing XML is to publishing what EPUB3 is to digital publishing. Like EPUB3, it:

- makes use of the XHTML-flavor of HTML5 as the core content document type. This provides for maximum compatibility with other content systems.
- makes use of the Dublin-Core and related vocabularies for publication metadata.
- provides extra vocabulary and semantics for dealing with the needs of publishing that are not met by HTML5 semantics.

Unlike EPUB3, Publishing XML considers the semantic needs of all publishing, not just digital or ebook publishing.

To learn more about the Publishing XML project, visit publishingxml.org.

Appendix B:
About the Author

Sean Harrison has been designing, implementing, and training people to use better publishing systems since 1999. These include systems to better integrate editorial and typesetting, print and digital production, ebook development for complex books and study Bibles, the production of app data, scripting Word and InDesign, and developing content-based web applications.

After serving with Tyndale House Publishers for 18 years, Sean launched Black Earth Group in early 2015 in order to help make publishing easier for a wider variety of publishers and authors. Sean loves publishing, he loves the technologies of publishing, and he loves helping his colleagues enjoy doing their work better.

Sean also enjoys photography, writing, and playing music with family and friends. Sean lives in northern Illinois with his amazing wife Lauran and their daughters.

Twitter: @saharrison

Websites: seanharrison.org (personal)
blackearthgroup.com (business)

Notes

Writing Tools

1. I should clarify that, when I say "plain text," I am talking about Unicode text in general and UTF-8 encoding in particular rather than ASCII or Latin-1 or any other lesser encoding. "Plain text" used to mean ASCII, but nowadays it means Unicode text as opposed to XML or some other "rich format." The ecosystem has matured to the point where all text should be edited in Unicode and stored as UTF-8, because this is the encoding that the software industry and the publishing ecosystem has rallied around, and all the major tools support it, including all of the "plain" text editors that I am recommending. To learn more about Unicode, visit http://www.unicode.org or https://en.wikipedia.org/wiki/Unicode. UTF-8 is explained in clear terms at https://en.wikipedia.org/wiki/UTF-8.
2. October 12, 2016.
3. See https://en.wikipedia.org/wiki/Markdown.

Writing and Editing in Word

1. On Windows, hold down Ctrl+Alt and then press 1, 2, or 3 in order to make the current paragraph Heading 1, 2, or 3. On macOS, hold down Option+Command and then press 1, 2, or 3.
2. You can download the template, with instructions, here: https://github.com/BlackEarth/word-manuscript.
3. Excellent, production-quality Greek and Hebrew fonts are available for free from SBL (Society for Biblical Literature) and SIL (Summer Institute of Linguistics):
   SBL Greek: https://www.sbl-site.org/educational/BiblicalFonts_SBLGreek.aspx
   SBL Hebrew: https://www.sbl-site.org/educational/BiblicalFonts_SBLHebrew.aspx
   Galatia SIL (Greek): http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=-GalatiaSIL
   Ezra SIL (Hebrew): http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=ezrasil_home
4. I say *should* because the production specialists should know how to do this successfully without any extra work on the part of the author or editor.

   Unfortunately, the most common note format, endnotes, is not supported well by the most common page layout program, InDesign. The solution to this conundrum is to avoid using InDesign's Word import and instead to use an XML-based workflow to convert Word manuscripts to InDesign. One such workflow is the Publishing XML workflow (see http://publishingxml.org). See further the chapter on production workflow. {TODO}
5. For a short history of the ZIP file format, see https://en.wikipedia.org/wiki/Zip_(file_format).
6. Microsoft Office: https://en.wikipedia.org/wiki/Microsoft_Office#File_formats_and_metadata.

7. Libre Office uses OpenDocument as its standard file format: https://en.wikipedia.org/wiki/OpenDocument.
8. EPUB: https://en.wikipedia.org/wiki/EPUB.

Typesetting with InDesign

1. At some point I should and probably will create a script to do all of this work automatically, so that after placing a Word document users can run the script and have nicely automated endnotes. Or, you could write such a script and contribute it to the bookgenesis project at http://bookgenesis.org.

Digital Pre-Production with InDesign

1. I suggest redefining this variable's format (Type > Text Variables > Define > Output Date > Edit > Date Format) so that it includes both the date and the time of the output, such as: MMMM d, yyyy h:mm a z
2. It is also possible to deal with video and audio by similar methods to images. {TODO}
3. Making use of this Note requires an export process that recognizes it. {TODO}
4. You might wonder why we have included the `pub:` prefix in the `<pub:include/>` and `<pub:section-start/>` tags. The reason is that, while the `<img/>` tag is defined in HTML and is the default, the other tags are not in HTML but in the Publishing XML namespace, represented by "pub:". See http://publishingxml.org for more information.

Archiving Content and Projects

1. Users don't necessarily see the benefit of these log messages right away, so it requires persistent training to get everyone to write descriptive log messages all the time.
2. I recommend setting up the server to use *https* and "basic" authentication; this is a simple setup that nevertheless provides good security.
3. https://github.com/seanharrison/professional-publishing
4. For this to work, the Apache configuration for the Subversion repository in question has to contain the following line:
   ```
   SVNAutoversioning On
   ```
5. davfs2 should be available using your package manager, or you can get it from http://savannah.nongnu.org/projects/davfs2