RWS

Automating publishing workflows through standardization

XML publishing with RWS

This white paper describes the use of XML standards to manage styles in publishing and how XPP® supports standards-based publishing.

XPP is an adaptive publishing engine that formats structured data into consumable content for print, PDF or virtually any other format for output.

Table of contents

Automating Publishing Workflows through Standardization

Publishing is Evolving

At one time, publishers worked in a well-defined, specialized area. Their job was to format content for a single print channel, delivering primarily PostScript or PDF. Although the Information Technology (IT) function was entirely outside of the publishing process, they were responsible for providing the computing environment that publishers used.

Unfortunately, this approach led to little interaction between Publishing and IT. The available computing standards did not directly support publishing processes – publishers focused on content preparation and formatting, while IT programmers and computer scientists developed corporate infrastructure and data standards.

Today, the advantages and wide adoption of structured content and web-based technologies are well known. As a result, the job of publishing organizations has grown to include delivering content formatted for multiple channels. These channels include PDF for online delivery, HTML for browser-based delivery and portable devices with customized, end-user interfaces.

Multi-channel delivery is now expected. This expectation challenges content providers to build content in a form that can supply all of these channels and has forced publishers into a new working relationship with IT – which is a good thing for content consumers.

Publishing is also evolving. New tools, smart apps, powerful computing environments, the internet and Cloud computing are changing the way content is created, reviewed, proofed and formatted.

In the past, the computing environment was simply there to support the publishing process. Today it is part of how it happens and what is produced.

Finally, the value proposition of publishing is changing. Can a publishing group elevate its value to an organization by leveraging tools to maintain strict formatting, or by investing in ways to exploit content reuse through structured content schemas? Is content inherently more valuable if delivered faster or more accurately? How can cycle times be reduced while demands for content over multiple channels increases? How can you keep costs down and still deliver quality? Is it possible to take your publishing organization from a cost center to a differentiator?

Automation is the Answer

The answer to these questions, not surprisingly, requires automation – but automation alone is not enough. For some time now, we've seen a variety of computing solutions come to market with many different editing, formatting and management applications/tools and numerous new ways to create and process content.

While these tools have helped with some processes and performance is better, most publishers have not fully achieved the promise of these technologies.

With the advent of XML, the IT community and publishers were finally working on the same page – the opportunity for publishers to fulfill the promise and solve their multi-channel delivery challenges while getting the productivity gains needed to remain viable and competitive.

Computing environments also continue to emerge using service-oriented architectures (SOA) and Web Services, making publishing processes accessible to a wider audience of content creators. SOA is a model for how applications interoperate as part of a business solution.

Web Services are how these applications or services interact via the web using a set of standards to define what services are available and how they communicate with each other.

As the migration from interactive desktop to batch publishing continues, the need to automate supporting content delivery functions increases – publishing becomes an on-demand service that plays a more vital role in revenue-generating, customer-centric business operations, creating differentiation through publishing.

The ubiquitous nature of XML has expanded the publishing process. Structured XML content is often the norm for data creation and storage databases. The XML family of standards for transformation (XSLT) and formatting (XSL-FO) have matured enough to normalize publishing infrastructures. Both XSLT and XSL-FO offer ways to reorder or transform data and to specify and apply formatting information to XML content – but not without some drawbacks. Finally, the use of Web Services as a means of communicating and moving content between applications adds a whole new dimension to publishing workflows.

Publishing Models

Many publishing processes exist today, with most based on either a batch (push) model or an interactive (desktop) model.

The challenge of desktop publishing (DTP) is that the tools are most useful for creating one-off, graphically intense or low-volume documents. DTP workflows do not scale easily to longer, more complex publications.

Most desktop publishing systems are only optimized for single-user interactivity. This creates challenges of scale, efficiency and workflow as data sources, complexity and volume increase.

Batch tools are designed for higher-volume data publishing and/or processing larger amounts of data through pre-established routines and styles.

Publishing is becoming an on-demand service that plays a more vital role in revenue-generating, customer-centric business operations, creating differentiation through publishing.

Tools in this class support batch processing of XML, but typically can't provide the same content interactivity or the formatted result that you can with a desktop system. They also often have limited processing power or automation (or both). In addition, support via standards may not be very robust (only limited parts of the standard are applicable) or proprietary (they do not support the standard or use closed methods to do so).

Another challenge in the batch model is that many systems that provide automation do not always produce high-quality publications with formatting or pagination errors appearing in the output. In these cases, creating an approved "final" publication often requires multiple re-batches.

Each re-batch then processes the entire document again, which fixes the known issues but often introduces new formatting problems. Sometimes resolving issues can involve repeated changes to document styles. Iterative style manipulations focused on individual document situations can lead to style-management inefficiencies and inconsistencies.

To address this, RWS incorporates the most efficient publishing models with the best of both batch and interactive models – highly automated batch processing for complex documents combined with interactive tools and single-page reprocessing for overrides, or less structured documents.

XPP was designed to overcome these challenges and expand opportunities for publishing operations. XPP supports both batch and interactive models while providing robust support for XML and other available standards.

Publishers and IT Leverage Standards

While desktop publishing is still commonly used for one-off or low-volume work, corporate or enterprise publishing jobs are commonly performed by service bureaus or by in-house hosted publishing operations, often with IT involvement.

The key to developing an effective, business-focused publishing process is to create an automated environment where tasks take place with little or no manual intervention. To build this environment, publishing and IT departments use standards-based business systems to create and deliver publications.

This goal of creating more effective, integrated, business systems and processes often leads to the following conclusions:

- Service-oriented architectures provide the most options for standards-based system integration and interaction.
- User interaction with and between applications needs to be more fluid and consistent. Web Services provide the best foundation for organizations to tailor the user experience by profile, skill set or role across multiple applications.
- User interfaces designed for productivity help to distribute publishing functions, enable more people to contribute to the content-creation process, and allow both experts and consumers to create their own personalized publications.
- Corporate-wide standards on the format and presentation of published deliverables improve document usability and brand identity, as well as reduce development, support, training and maintenance costs.
- Implementing a publishing system as an on-demand service aids in revenue-generation, customer-centric business operations and product differentiation.

Where XSL-FO Falls Short

XML, the standard most responsible for publishing normalization, now has an ecosystem that encompasses everything from content structure and markup to systems communications protocols. Today, there are a number of systems for creating, managing and storing data in XML format. And thanks to the web, systems effectively communicate using XML standardized messages.

Through consistent data and open-system interaction, content reuse and sharing are expanding rapidly. To optimize reuse, however, the industry needed more than just XML. It needed standards focused on information transformation and presentation in a multi-channel delivery environment, resulting in standards to manage style sheets.

A popular style-sheet standard, XSL, generates XSL-FO and leverages XSLT for transformation. However, not all procedural-language programmers are proficient with XSL. In addition, XSL only works for XML documents. If your data is in another format, like structured ASCII, it needs to be merged with other data to generate an acceptable format.

To address this, RWS goes beyond XSL and uses cascading style sheets (CSS) to open up print publishing to any web developer. However, extensible stylesheet language formatting objects (XSL-FO) is still often used to describe style and formatting information for XML content. XSL-FO is a language for expressing the output format of XML documents and is conveyed in a well-formed XML instance.

XSL-FO can be applied to many types of publishing, such as statements, reports, marketing materials, basic technical documentation and other, more complex automated publishing outputs.

XSL-FO processing flows adhere to the batch model, in which data is more complex and formatting more important. Unfortunately, this workflow frequently requires many re-batches of the entire publication. As a result, XSL-FO can fall short for page-based decision requirements such as footnotes, image position, stacking, balancing and facing pages.

For high-end or sophisticated publications such as legal publishing, loose-leaf output, journals, or reference materials, this causes many issues. Because XSL-FO necessarily performs a transformation (taking from the source XML to create a new XML document), it provides no interactive page editing ability and gives publishers less control over fine typesetting parameters. Although you could edit the resulting XSL-FO document, it is very inefficient and doesn't allow for changes to a single page – everything is reflowed.

In addition, batch composition using XSL-FO for composing pages, sizing pages, locating figures, etc., requires multiple steps and provides fewer controls than XPP does right out of the box. XSL-FO does not provide the same level of typographic sophistication as XPP for things like vertical justification between lines, alignment of financial data, etc.

Figure 1. A sample XSL File

XPP – An adaptive publishing engine

XPP (XML Professional Publisher) is an industry-proven, standards-based composition, transformation and rendering software. These capabilities are derived from its adaptive publishing engine. It provides robust functionality for complex, automated publishing in technical documentation, commercial, legal, journal, financial and pharmaceutical publishing markets. Many leading corporations use XPP to process and transform XML and other structured content into high-quality PDF and print output.

XPP provides a single application capable of supporting new and legacy approaches to implementing style sheets. In XPP, you can define style rules and have multiple passes to compose a page. For example, if a graphic or text call-out is on the same page as another graphic or text call-out, you have multiple options to get the desired results.

XPP offers both batch (push) and interactive (desktop) capabilities and has strong, continuously evolving XML support. Based on this versatility, XPP is often deployed as an enterprise publishing tool and is used by many organizations as a formatting and print "service" for content creators to publish high-quality composed documents and other related outputs and formats.

Creating and delivering accessible content is a desire and requirement for many organizations. XPP provides for the streamlined creation and publication of PDF/Universal Accessibility (PDF/UA) documents to comply with US Section 508 regulations and ISO requirements.

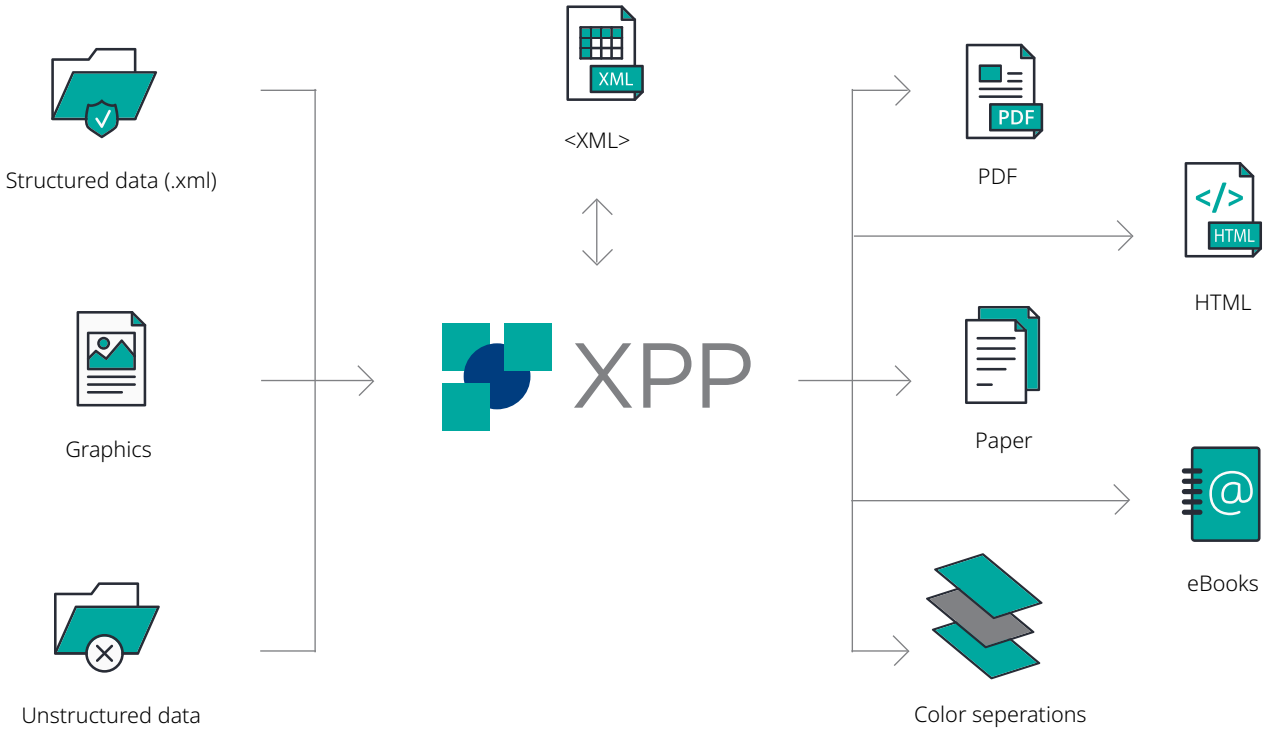


Figure 2. The XPP Data-flow and Processing Model

XML Content and CSS in XPP

XPP supports XML content with style and page set-ups optimized for XML tagged data, attributes and structures. It uses Cascading Style Sheets selectors, optimized for XML-tagged data, to determine context and can parse both valid and well-formed documents as part of the composition. XPP has provided this level of functionality in production for years, at some of the most complex publishing environments in the world.

CSS is maintained by a working group which was created by the World Wide Web Consortium (W3C). The working group describes CSS as, "a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents."

Although it is still simple, many organizations have pushed the standard to include HTML, XML, PDF, etc. CSS is compatible with most browsers and many applications and does not require a specific text editor for authoring, giving more flexibility to the publisher.

The default or implied processing model identified in the standard brings together an XML instance and a CSS stylesheet, processing them through XPP to create a file containing both content and style information to produce paginated output.

XPP's adaptive processing engine supports the CSS processing model with the addition of interactivity and robust processing capabilities required for complex and/or high-volume publishing environments. XML publishing processes benefit from XPP's support for XML content and CSS style statements to drive the SXPP composition engine.

CSS offers many advantages to XSL-FO. CSS enables users to automatically compose and render more complex documents that are typically not supported in FO (or by FO engines), offers the interactivity often required for graphically intense documents or for page corrections, and can support the import, composition and export of valid XML files from the XPP system.

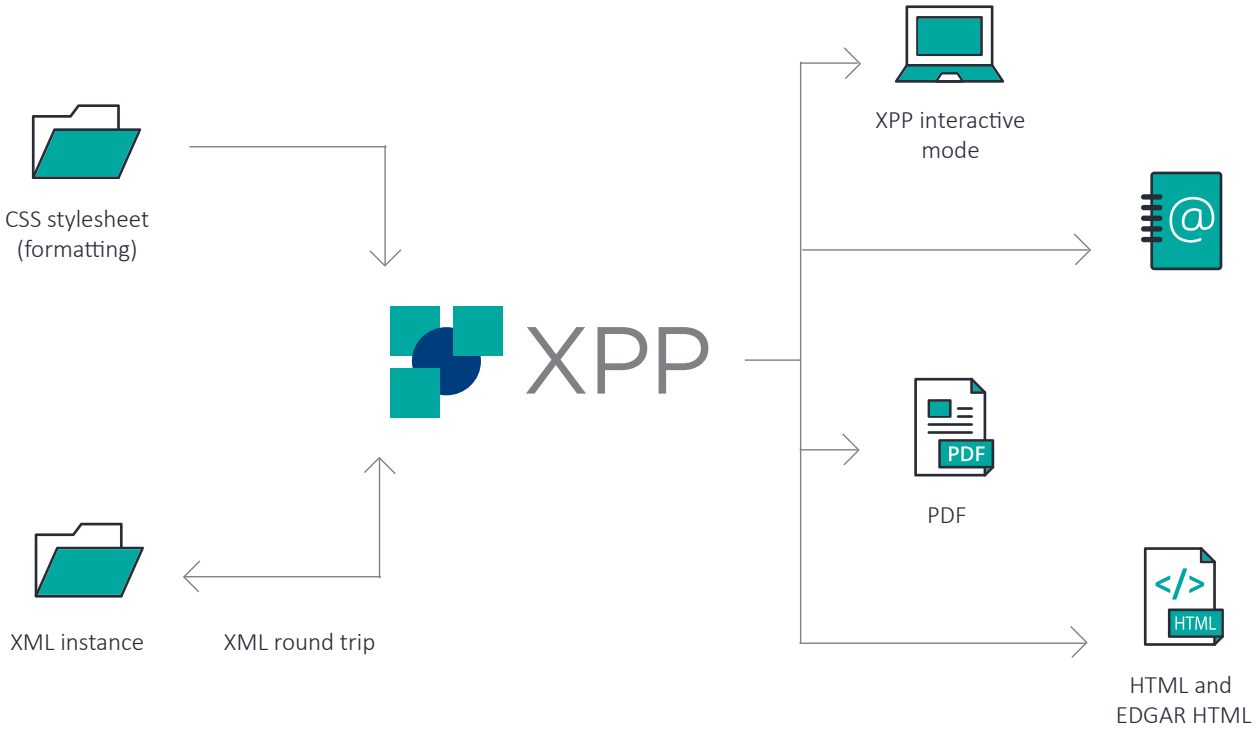
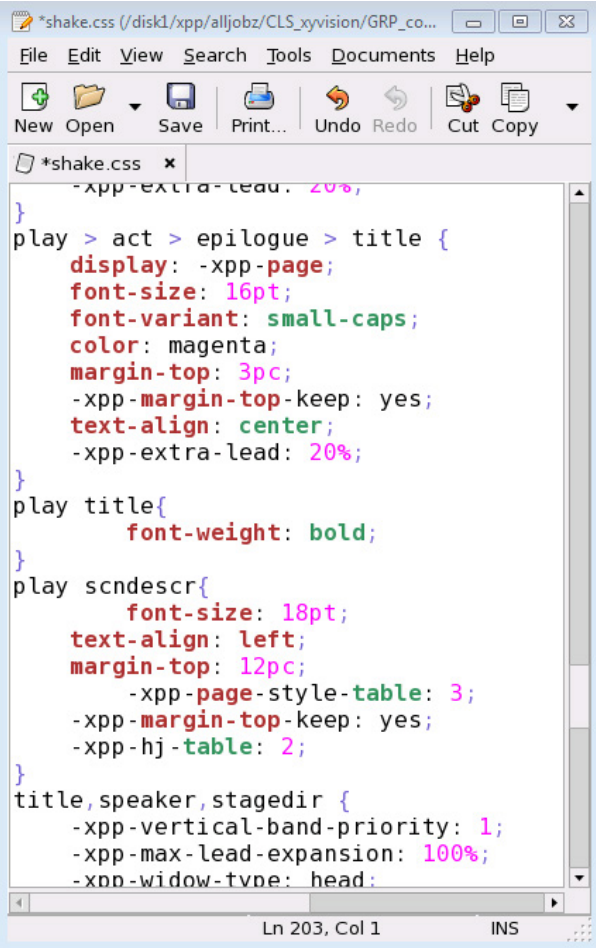


Figure 3. The CSS Processing with XPP Batch and Interactive Capabilities.

XPP support for CSS is built on the application's core XML content, its proven style management and its processing model. Using CSS, XPP preserves the original content, while using CSS style statements to generate the selected style format to create paged output. By doing this, the original content stays the same and benefits from the cascading nature of CSS to apply specific style rules with no transformation, unlike XSL-FO, which performs content formatting and transformation together.

In this processing model, XPP maintains a separation between content and formatting, although content transformations can be done before loading into XPP. Using CSS, XPP uses style statements to determine the document XML tag hierarchy to apply styles. By not performing a transformation, the original XML is unchanged and allows for editing and seeing the edits "as is." With XPP, style statements are applied to flowing XML that retains the content, but now in page form. This is how XPP allows for interactive editing, reapplying styles to one page, to one block, to one line, or to one character.



Leveraging Web Services

When RWS introduced the XPP Web Services Development Kit, the core publishing power of its adaptive publishing engine became available to a wider range of users and operations. Through our Web Services interface, users can use their browser to proof, modify and publish documents from virtually any location, reducing cycle times and enhancing production quality.

This Web Services layer makes it easy to deploy XPP as a publishing service in service-oriented architectures for a variety of IT deployments and hosted applications. The Web Services layer also enables the deployment of configurable or customizable browser interfaces to XPP.

Conclusions

The world of publishing continues to evolve. The inherent connectivity and standardization of the web have created many ways to offer software, services and publishing automation tools to a wider variety of users and operations.

RWS continues its commitment to automated publishing workflows by supporting traditional, current and future publishing models like publishing-as-a-service for XPP (cloud). We'll also continue leveraging XML and the related XML working-group solutions and the XML family of standards to support efficient business practices and publishing models. Our vision for standards-based publishing is:

XML for content and output of page representation (export)

- HTML
- EDGAR HTML
- ePubs

CSS for formatting

- Replaces proprietary style development
- Pre-configured style sheets
- DITA/S1000D/Web and other structured-content sources

Support for multiple languages

- Hyphenation support

Flexible pagination controls

- Footnotes
- Image control (position)
- Stacking/balancing
- Facing pages, etc.

Web Services for integration

- XSLT for transformations (applied before loading into XPP)
- Content preservation

PDF/UA capabilities

- Tagging
- Bookmarks
- Annotations

RWS's support of standards, combined with the capabilities of our XPP adaptive publishing engine, provide tremendous value today – and the foundation to evolve and expand publishing automation solutions for the future. XPP provides answers for style management, with style standards such as CSS and its publishing strength, it is the most powerful standards-based batch and interactive publishing product available.

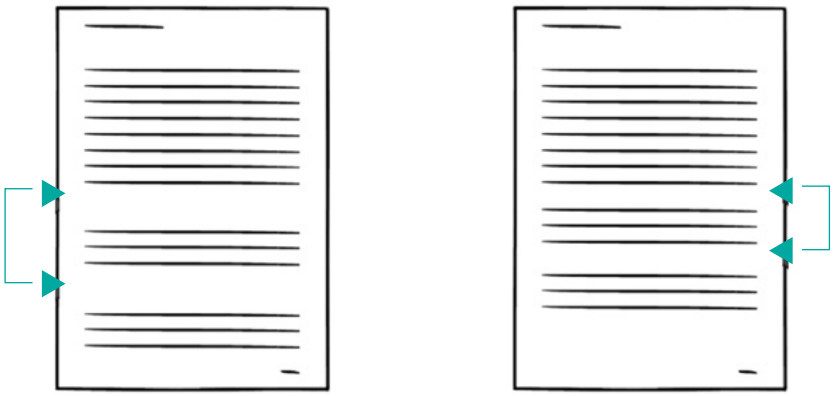
Find out how RWS's commitment to high-performance publishing and XML standards can provide value to your organization. Visit rws.com/xpp

or email askanS1000Dexpert@rws.com

Appendix: A sample comparison of features implemented in XPP combined with CSS versus XSL-FO

Vertical Justification

XPP	XSL-FO
White space is controlled through feathering and priority. As a result, the page can be filled out evenly.	White space is pre-defined and cannot be changed.



Continuation

XPP	XSL-FO
The border of the box doesn't have a bottom line on the first page. In XPP, you can indicate that a note continues on the next page.	In XSL-FO, you have to have borders on all sides. The work-around is to put text in a table, but processing and maintenance become more difficult.

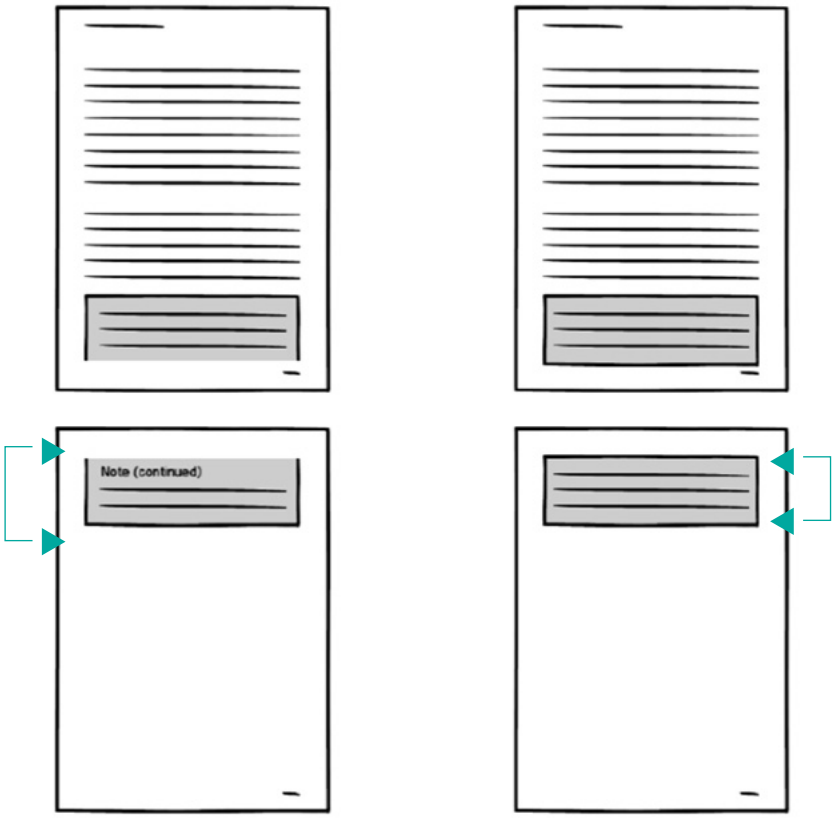
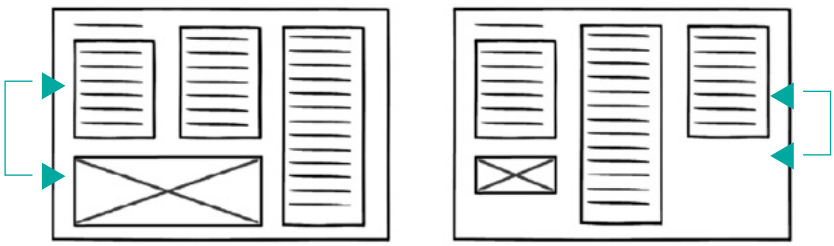


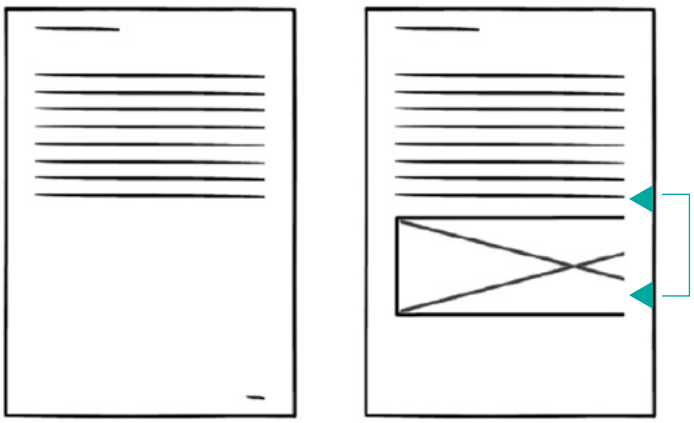
Image spanning multiple columns

XPP	XSL-FO
In XPP, the image can span multiple columns easily.	In XSL-FO, you can only resize the image to the width of the column.



Appendix: A sample comparison of features implemented in XPP combined with CSS versus XSL-FO

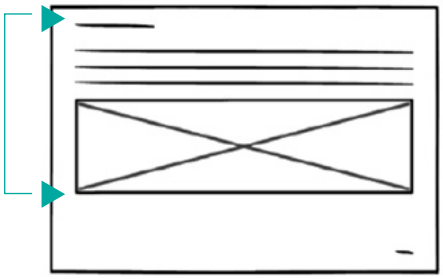
Rotating a page based on the size of an image



In XSL-FO, if an image is too big for a page, the image will be cut off. It is not possible to define rules to use a different page orientation.

Headers can be repeated, as they are independent from page sequences (text registers).

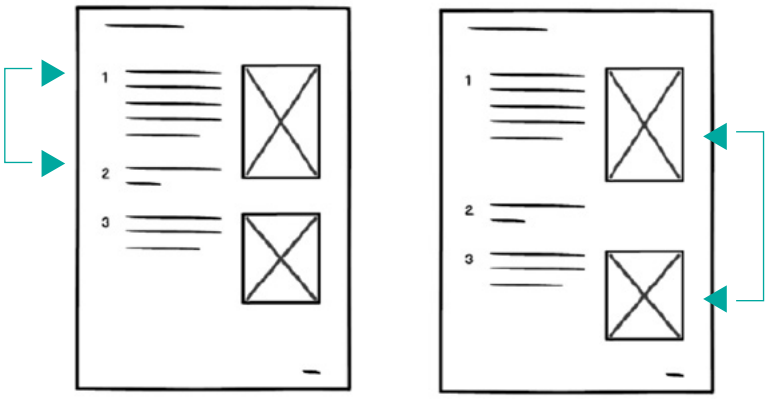
InL XPP, you can calculate the width of an image. If the image does not fit the page, you can switch the next page into landscape mode.



The only way to do this in XSL-FO is to define a different page sequence. Based on some semantics of the image (defined in the XML), you could use this landscape made page sequence. The issue then becomes getting the correct border into this new page sequence.

Image Alignment

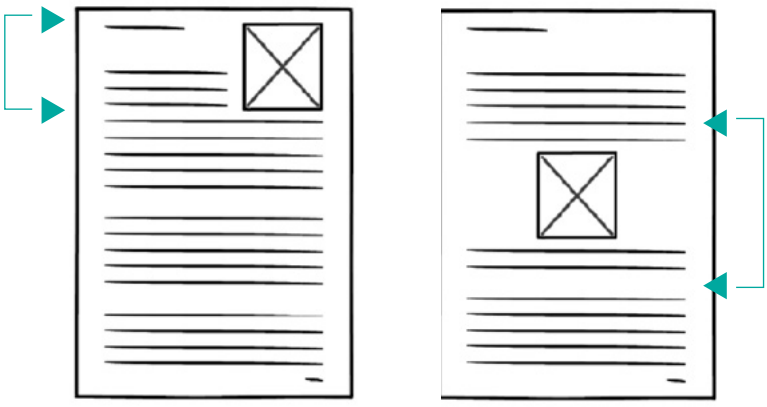
In case the image is larger than the text in the first step, you can have the image overflow into the second step.



In XSL-FO, the only way to do this is to use a table to render the information. This complicates style development greatly, as you are misusing a table to achieve the correct layout. It is then impossible to have the first image overflow into the second step.

Repositioning Images

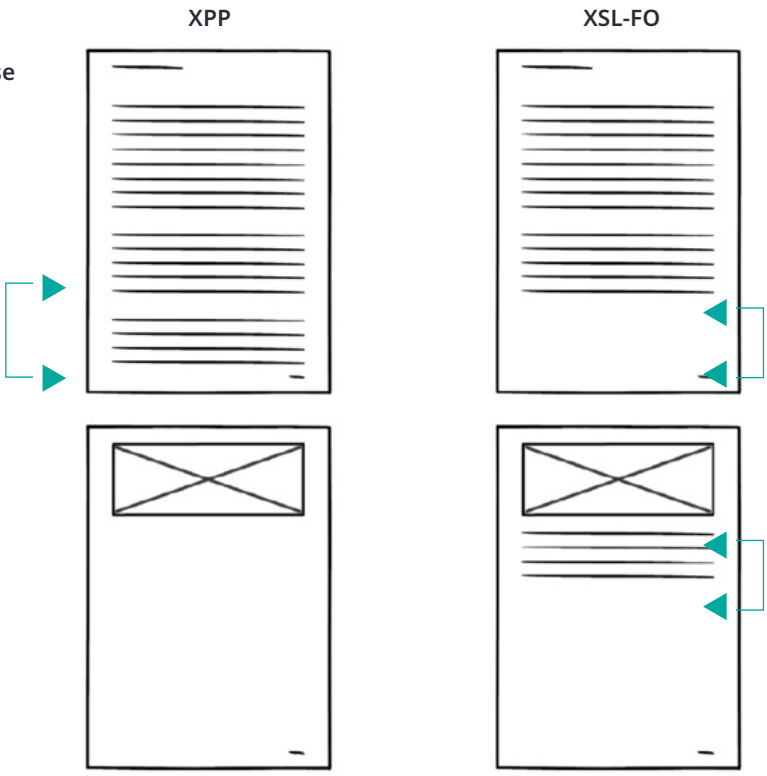
In XPP, you have the flexibility to reposition images into four locations: Top, Bottom, Center, Left-Right Center.



In XSL-FO, this image is always rendered in the location where it resides in the XML.

Appendix: A sample comparison of features implemented in XPP combined with CSS versus XSL-FO

Repositioning text because image overflowed

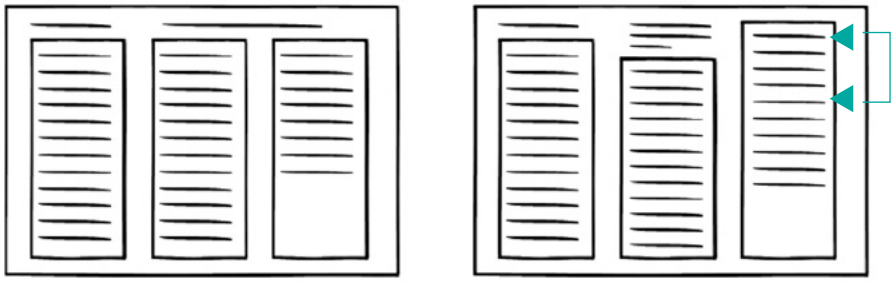


The XML located the text below the image. In XPP, you have the flexibility to fill out the page and show the text before the image.

XSL-FO will produce unwanted whitespace on the preceding page.

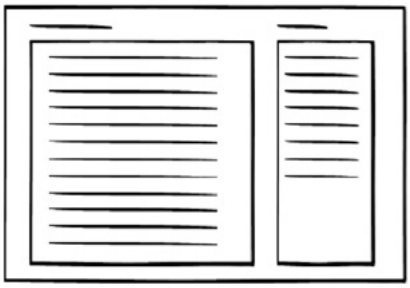
The text will be placed after the image.

Titles can span multiple columns



In XSL-FO, the title needs to stay within the boundaries of the column. A long title will always wrap, pushing the rest of the text down.

Columns of different sizes



In XSL-FO, the only way to get columns of a different size, is to put everything on the page into a table.

You can only specify a column gap, but not the size of the individual columns.

Footnotes and columns