Unlock the Business Value of your Content: Move from Unstructured to Structured FrameMaker

A Step-By-Step Migration Guide to Structured Content

Bernard Aschwanden
President, Publishing Smarter

Unlock the Business Value of your Content: Move from Unstructured to Structured FrameMaker

A STEP-BY-STEP MIGRATION GUIDE TO STRUCTURED CONTENT

When a business decides to write content using Adobe FrameMaker it is done, in part, to free writers from many of the difficulties associated with other software. Issues may include:

- Working with large volumes of content
- Managing complex numbering systems
- Meeting online or print design and layout requirements
- Publishing to Responsive HTML5 (plus apps, ebooks, and PDF), or
- Other reasons to retire *just* a word processor and use a professional communications tool!

Go further! Unlock your content from format and form. Free yourself to focus entirely on your content and audience. Define the semantics of your content and unlock its value in any format, to any audience, on any device.

This whitepaper explores how you can take the value of your content and add the functions and features that structured content delivers. We'll provide hands-on exercises to give you basic instruction on analyzing content and converting it to structure. Along the way you'll learn more about DITA (the Darwin Information Typing Architecture) as well.

TABLE OF CONTENTS

WE WROTE THIS FOR: You! An experienced user of FrameMaker, you are generally comfortable using the Paragraph, Character, and Table Designers, and are considering a migration to structured formats such as DITA or other XML standards. Welcome aboard.

CHALLENGE: Developing a structured environment requires the migration of legacy content, while also learning new tools, and related structural concepts. In many cases there is also a need to develop supporting content for importing content to (or exporting from) FrameMaker in the native XML code. Template and format changes, structural needs, and analysis of content all take time. We wanted to do something to save you some of that time.

Moving to structured content needs hands-on involvement, and with the information in this paper you'll have a chance to see how it can be done, practice step-by-step examples and decide how your content can best be migrated to structured formats.

ABOUT US: Publishing Smarter (**www.publishingsmarter.com**) helps companies communicate. We develop and implement content strategies This includes creating, managing, and distributing content from one source to many outputs. We've worked with hundreds of companies to help develop FrameMaker templates, migrate content to structure, analyze content, develop a content strategy, choose a CCMS, and deliver training.

Overview

Many companies are looking to migrate from unstructured to structured content. By writing a (very detailed and lengthy) document such as this our goals are to explore and discuss:

- Migration to semantically marked up content (**Section 1: XML and structured content**)
- Use of Adobe FrameMaker as the tool to do this (**Section 2: Benefits of structured FrameMaker** and **Section 3: Working in a supported structured authoring environment**)
- Specific step-by-step hands-on examples so you can learn about unstructured content migration to structured content using FrameMaker. This is explored in **Section 4: Hands-on development**

We'll include examples based on work we have done with clients, but we'll also give you enough guidance that you can get started on your own. While the samples we include are simple, the ideas we explore are clear and can be expanded to your specific content. Having worked with hundreds of clients in the past 20 years it's safe to say that while their content is unique, both unstructured and structured content should always be created using best practices. We've formalized a lot of that thinking into the following document and hope it's helpful as you start to take the journey to structured content.

You can even dive right in and get hands-on (without ANY background content) by jumping directly to **Section 4: Hands-on development**!

Section 1: XML and structured content

Extensible Markup Language (XML) is a vendor-neutral, open format managed by the World Wide Web Consortium (W3C). HTML is also vendor-neutral, but is a pre-defined set of tags that primarily focus on the display of information on devices. Both have tags enclosed in angle brackets (such as <**b**> for bold), are relatively easy to learn, and generally are stored in a text-only format.

It is crucial though to recognize that, aside from their superficial similarities, HTML and XML are used very differently. HTML is for online display of content using a fixed set of tags which, by and large, have no meaning beyond format. XML is for describing information in a semantic and meaningful way, and uses an extensible set of tags which define a logical structure.

Differences between HTML and XML

XML has core differences with HTML that makes XML easier for people to understand, and simpler to work with for computers. The language is human-friendly; the information semantically meaningful. You can create your own tag names if you wish (for example, it may make sense to organize the bestselling books of all time and include information about the genre, author, publish date, title, or numbers sold) to provide more meaning to the content if and when manipulated by people. Software can quickly process XML files. Parsing a plain text file is done in a split second, and can be done without having to first deconstruct the parts of a document.

EXAMPLE: Consider the content coded in HTML and XML seen in **Table 1**. We've made it easier to follow by formatting **the tags** and *the content* so they stand out from each other. It's easy to see the HTML structure on the left, but difficult to extract further meaning from it. If you read the HTML and focus on the tags, this could be written about anything. The sample shows a side-by-side comparison of HTML and XML using some work we've done with an online book retailer. You can see how much more usable the structure of the XML content on the right is.

Table 1: Comparing HTML and XML content

HTML example	XML example
<html> **<head><title>***Top Sellers* **</title></head>**	**<booklist>** **<title>***Top Sellers***</title>** **<body>**
<body> **<h1>***Don Quixote***</h1>** **<p>***Miguel de Cervantes***</p>** **<p>***500 million***</p>** **<p>***1605***</p>**	**<book genre="***fiction***" published="***1605***">** **<title>***Don Quixote***</title>** **<author>***Miguel de Cervantes***</author>** **<sold>***500 million***</sold>** **</book>**
<h1>*A Tale of Two Cities***</h1>** **<p>***Charles Dickens***</p>** **<p>***200 million***</p>** **<p>***1859***</p>**	**<book genre="***fiction-historic***" published="***1859***">** **<title>***A Tale of Two Cities***</title>** **<author>***Charles Dickens***</author>** **<sold>***200 million***</sold>** **</book>**
<h1>*The Lord of the Rings***</h1>** **<p>***J. R. R. Tolkien***</p>** **<p>***1954-1955***</p>** **<p>***150 million***</p>**	**<book genre="***fantasy***" published="***1954-1955***">** **<title>***The Lord of the Rings***</title>** **<author>***J. R. R. Tolkien***</author>** **<sold>***150 million***</sold>** **</book>**
... **</body>** **</html>**	**...** **</body>** **</booklist>**

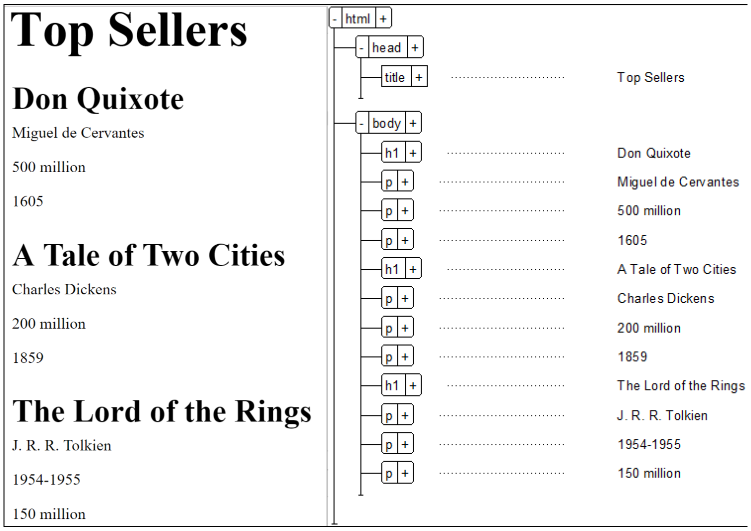
The HTML example lists each book as a heading followed by some paragraphs. No specific grouping exists at a logical level beyond the <**body**> of the content. It's difficult to uniquely identify the structure and extrapolate meaning without reading the text inside the markup. Lastly, and perhaps most importantly, there is virtually no way to uniquely identify the information by function. You may identify the <**h1**> as the title, but the following content is a collection of paragraphs (the <**p**> elements), and these are inconsistent in their content.

If you look carefully you may even catch the error in the last book element (The Lord of the Rings). Unlike previous entries someone entered the year first, and followed it by the number sold (all others have the numbers sold first, and then the year second). Even risks associated with human error can be reduced using XML.

However, it is far simpler to comprehend the semantic and meaningful structure of human-readable XML tags. Spacing between the paragraphs is only added to make it a bit simpler to compare "apples to apples" in the content.

The other major difference is in the output that can be generated. Comparing **Figure 1** and **Figure 2** shows how HTML, by default, is a linear product, but that XML can be adapted, re-organized, or even have some elements excluded.

Figure 1: HTML sample of structure with output and hierarchy



Top Sellers

Don Quixote

Miguel de Cervantes

500 million

1605

A Tale of Two Cities

Charles Dickens

200 million

1859

The Lord of the Rings

J. R. R. Tolkien

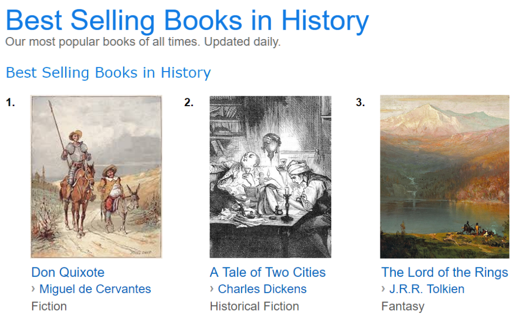
1954-1955

150 million

The web was driven by HTML code, but that code has no semantic structure, isn't database driven, and is used for just one purpose. However, when content needs to be created, managed, and published from one source, issues arise. One solution is structured, XML-based content. This follows rules and looks beyond basic tags like <**i**> and <**b**> (for italic or bold) and digs into the meaning of your work using tags such as <**wintitle**>, <**section**>, <**brand**>, <**stepresult**>, or any tag that you may decide to define. The options (and the benefits) are virtually limitless with XML.

Figure 2 shows that XML content can be transformed and, based on semantic markup, a specific layout or design applied for a given output. The logic used to define the order or the format of content can be driven by the elements <**title**>, <**author**>, or <**sold**> meaning that books can be sorted by the number sold, and formatted based on the element name.

Figure 2: Sample of one (of many possible) outputs supported by an XML source



Best Selling Books in History

Our most popular books of all times. Updated daily.

Best Selling Books in History

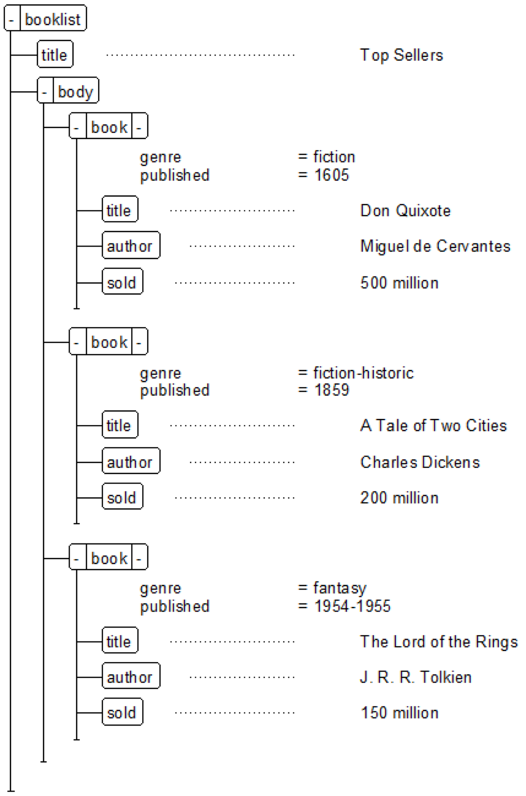
1. Don Quixote
› Miguel de Cervantes
Fiction

2. A Tale of Two Cities
› Charles Dickens
Historical Fiction

3. The Lord of the Rings
› J.R.R. Tolkien
Fantasy

Lastly, exploring the content structure, you can see a clear hierarchy for all booklist information. Each book has specific information about the genre or when it was published, as well as the title, author, and volume sold. The markup tells the story of the type of information you see.

Figure 3: XML structure sample hierarchy



Looking at the XML example, you can immediately understand and use the content even if there is no format assigned to it. The semantic markup tells us, as humans, that this is a list of books. The genre is identified early, as is the publish date, and content is organized to simplify finding the title, author, or how many were sold. There is also information about the <**book**> in attributes (this information is often called metadata). The attributes define the specific genre and the year published. You can quickly create a list of top-selling authors from this content, or parse it for all books selling between 10 and 25 million copies, which were published between 1983 and 2004. In short, the content in the XML file is usable beyond just displaying on the web. It has meaning.

XML uses well-formed and valid structure

Well-formed, structured content means predictable results when software is used to process files. The processing could be for system-to-system communications between devices or services (for example, a bank creates an app that communicates between your phone and a terminal used to pay for goods), or for FrameMaker to open the document and allow review or edits, or even for conversion through automated systems from one source of structured content to another output (such as automated conversion to PDF or web and mobile content).

Structure requires well-formed content where all tags are opened and closed properly. For example (remember, we've formatted **tags** and *content* to stand out from each other):

- The following sample contains text only and has clear start and end tags that indicate it is a paragraph. The paragraph is short, self-contained, and has no additional structure within it. It properly marks up content and opens and closes a tag without overlap.

  <p>*When working with XML it is important to get to know a bit about the structure, even if you don't want to know about all the code and markup.***</p>**

- The following sample contains text and has markup to indicate it is a paragraph. However, it also contains a window title (the name of a dialog box) that appears when working.

  <p>*Structured content is similar to unstructured content when you work with it. You can open, edit, and close files. Dialogs such as the* **<wintitle>***Open***</wintitle>** *may have additional options to support the XML content though.***</p>**

- The following invalid sample has one element overlapping another element. This is not valid when working with XML and therefore is very difficult to even create for a demonstration!

  <p>*Invalid content exists when one element has a start or end tag contained within another element and* **<i>***this must be avoided at all times.***</p></i>**

In the last sample the tag **<i>** is used to indicate italic content. However, it encapsulates the closing tag for the paragraph. For the content to be valid the closing tag (the **</p>** content) would have to be moved to the right of the closing tag for the italic. While HTML may allow this type of content, structured content that must adhere to the rules of XML will not. Quality software should never allow such invalid markup to be created.

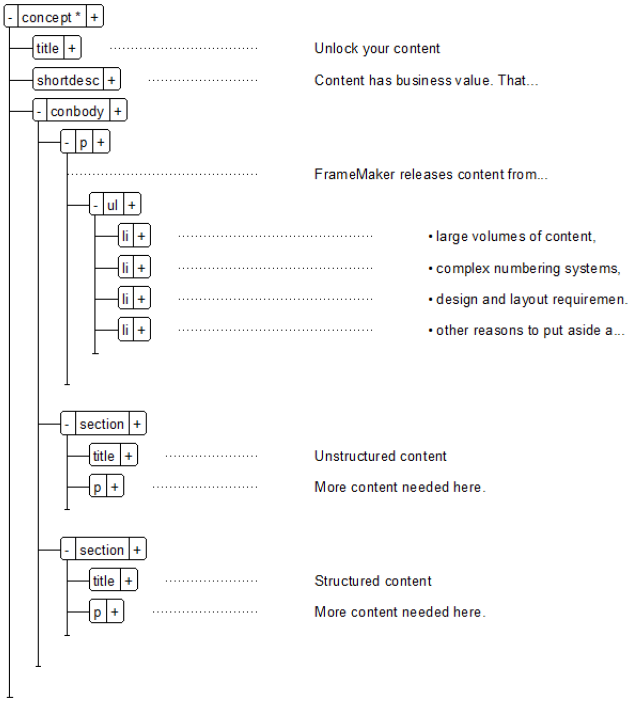
XML is a broad standard to encode content. There are however a range of options on how that content is encoded. All XML structured content is tagged with markup. Technically, you can define your own and we have many clients who do so. FrameMaker supports many standards (DITA, S1000D, DocBook, etc) with built-in templates and functions so you don't have to do nearly as much work if you use one of these standards. To keep things simple (and to allow you the option of expanding your understanding of the samples we use) we will work with DITA (a leading industry standard) throughout our content. This means all the reading you do will provide you a better understanding of DITA.

EXAMPLE: **Table 2** is a valid structured XML document, written to adhere to the DITA standard. To enhance human readability *content that is part of the written document* has one format and **the tagged markup** another. Just like our previous examples of a list of books, this XML content has meaning based on the tags used.

Table 2: Sample of valid DITA XML content

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE concept PUBLIC "-//OASIS//DTD DITA Concept//EN" "concept.dtd">
<concept id="unlock-content">
  <title>Unlock your content</title>
  <shortdesc>Content has business value. That value is in the ability of people to learn what you
  offer, how you perform services, how they can perform tasks, understand concepts and ideas, find
  technical reference, or otherwise learn more about what your business does and offers.</shortdesc>
  <conbody>
    <p>FrameMaker releases content from difficulties associated with other tools. This may include:
      <ul>
        <li>large volumes of content,</li>
        <li>complex numbered systems,</li>
        <li>design and layout requirements, or</li>
        <li>other reasons to put aside a word processor and use a professional communications tool. </li>
      </ul>
    </p>
    <section id="unstructured">
      <title>Unstructured content</title>
      <p>More content needed here.</p>
    </section>
    <section id="structured">
      <title>Structured content</title>
      <p>More content needed here.</p>
    </section>
  </conbody>
</concept>
```

Figure 4: XML structure in a tree view


concept *
title
Unlock your content
shortdesc
Content has business value. That...
conbody
p
FrameMaker releases content from...
ul
li
• large volumes of content,
li
• complex numbering systems,
li
• design and layout requiremen.
li
• other reasons to put aside a...
section
title
Unstructured content
p
More content needed here.
section
title
Structured content
p
More content needed here.

Structured content gives you a view of the document that is based on the hierarchy and function of the content. This is a quick and easy way to see the relationships among the tags.

DITA and its relationship to XML

DITA (the *Darwin Information Typing Architecture*) is based on XML, and is one way to encode content. This could be seen as similar to the idea that there are standards for cell phones to connect with towers, but a company that makes a specific phone decides how to interpret the standard to make their phone work with the network.

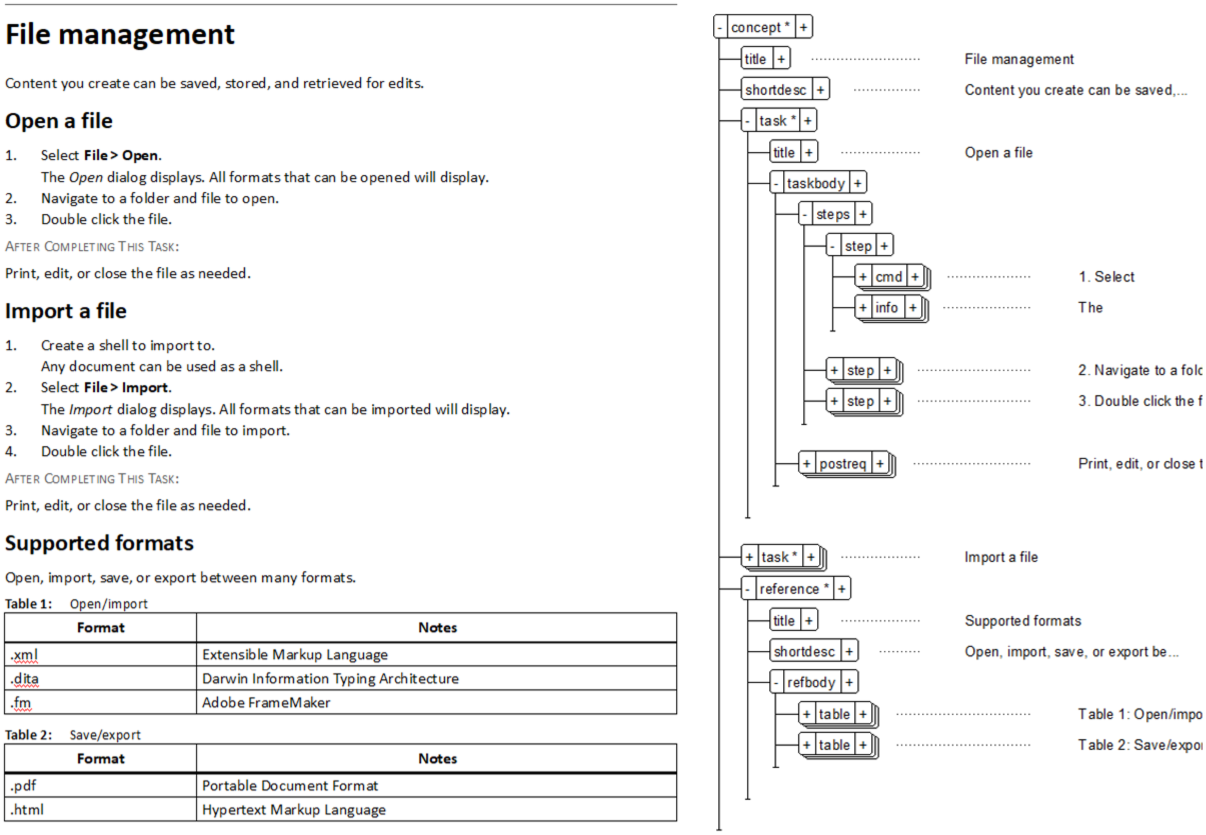
DITA organizes information into maps (similar to the idea of a FrameMaker book) and topics. Topics are smaller than chapters—usually made up of a mix of how-to information (task), technical information (reference), and overview or 'what is'-type of information (concept). Other topic types available and a full and rich discussion about DITA can be found in numerous online forums, books, and conferences.

Capturing information hierarchy

Structured content is about the purpose of information, not its physical appearance. This adds value to content through the context of information, and the classification of the information. Word processing workflows expect part of the value of the information to be implied through the visual appearance. Not so with structure. Instead, the writer focuses on content hierarchy .

In structured content the relationships between elements matter. Content is not just paragraphs with arbitrary format assigned to them. The logic of the hierarchy makes it easier to identify information and define where it is within a document. This allows the format to be defined dynamically and ensures a logical structure that can be enforced and acted upon.

Figure 5: Comparing formatted content and logical structure



File management

Content you create can be saved, stored, and retrieved for edits.

Open a file

1. Select **File > Open**.
   The *Open* dialog displays. All formats that can be opened will display.
2. Navigate to a folder and file to open.
3. Double click the file.

AFTER COMPLETING THIS TASK:

Print, edit, or close the file as needed.

Import a file

1. Create a shell to import to.
   Any document can be used as a shell.
2. Select **File > Import**.
   The *Import* dialog displays. All formats that can be imported will display.
3. Navigate to a folder and file to import.
4. Double click the file.

AFTER COMPLETING THIS TASK:

Print, edit, or close the file as needed.

Supported formats

Open, import, save, or export between many formats.

Table 1: Open/import

Format	Notes
.xml	Extensible Markup Language
.dita	Darwin Information Typing Architecture
.fm	Adobe FrameMaker

Table 2: Save/export

Format	Notes
.pdf	Portable Document Format
.html	Hypertext Markup Language

EXAMPLE: The text and visual format of **Figure 5** could be represented on the right as a collection of paragraphs formatted as headings, body content, steps, table components, and so on.

In the structure view on the left the same document can be seen as a collection of semantically organized parts instead of as a flat list of paragraphs and tables. There are two clearly defined tasks with title and task body components. The task body contains steps, and each step in turn has logical parts. Each part of the structure serves a specific function and can be expanded or collapsed for viewing or reorganization, defined as required or optional, and dynamically formatted based on the unique template used and the hierarchy of content.

Embedding metadata in XML

Metadata is information about information. Sounds a bit odd, right? Metadata provides more detail about an element as you'll see in a moment. Any element has the potential for optional attributes which allow you to store more information.

The following two elements could be part of the same document but targeted to specific readers and may only appear in the output if a specific audience is reading the content:

- <**p** audience="novice"> (the element **p** is written for for an **audience** who is ***novice***)
- <**p** audience="expert"> (the **p** has an attribute to describe the **audience** as ***expert***)

Let's look at a few more elements with attributes and values used to define metadata:

- <**section** product="fm" version="2020"> (**section** has 2 attributes for a product and version)
- <**example** product="fm"> (this example is specific to just the product FrameMaker)
- <**step** platform="iOS"> (only applies to iOS)
- <**book** genre="fiction" published="1605"> (a book of fiction written a very long time ago)
- <**map** id="id_123" audience="pro" product="fm" version="2019"> (a map that guides the expert user through FrameMaker 2019)

EXAMPLE: A graphic in HTML may appear as: <**img** src="name.ext" height="75" width="200" /> which defines an image, identifies the image source and sets the height and width in pixels.

These attributes are basic metadata that further define the img (image) element. Metadata for elements could include many types of information such as those that define the audience to whom content applies, a specific product or platform, status of the content, uniquely identify specific steps in a task, or for a range of other functions.

Benefits XML offers

While XML is based on code (just like HTML for the web) there are many tools that simplify the view and allow authors to focus on content (just like HTML authoring for the web). We'll show you code so you can understand what is happening behind the scenes, but always remember that a professional author tool hides the code, provides the semantics, and allows writers to focus on their writing.

Extensible Markup Language (XML) is a set of configurable rules for content that define the structure of content, not the appearance. It is most often guided by a formal set of rules that specify what options can be considered in a given context.

One XML rule set many people in the content creation field use is DITA. This allows software and services vendors to start with a common baseline to develop applications and solutions that can be simpler to implement. DITA supports content creation in a topic-based structure and simplifies reuse, management, and publishing.

Consider a single structured hazard statement used in the product safety information, as it might be written in XML.

Table 3: Hazard statement sample

```
<hazardstatement type="laser3B" id="SafeT-1st-3B" last-update="20180123">
 <messagepanel>
  <typeofhazard>LASER RADIATION CLASS 3B</typeofhazard>
  <consequence>DIRECT EXPOSURE TO THE EYE CAN CAUSE BLINDNESS.</consequence>
  <howtoavoid>DO NOT LOOK DIRECTLY AT THE LASER.</howtoavoid>
 </messagepanel>
 <hazardsymbol href="laser.png"/>
</hazardstatement>
```

Ideally, it's written so that a human can even see the code and identify that it's a safety warning about laser radiation and has information on the type of hazard faced, what could happen to someone, and how to avoid the hazard. The look and feel can also be assigned to content based on the structure. This would allow, for example, the <typeofhazard> to be bold, <consequence>

red, and <howtoavoid> reversed (white text on a black background). Later, if for example, you decide the <howtoavoid> should be on a red background, you can update your template and reformat the look and feel of every <howtoavoid> element immediately!

Enforcing consistent organization

Unstructured files allow authors flexibility in creating content. Consider a task in which steps contain a range of instructions. Identifying which content is first, last, or in any specific order is left to a style guide, author training, trust in the process, and human memory. In many cases, an additional review of all content is required to ensure the style guide is consistently followed and both the sequence of steps and information to support steps is organized according to standards. In structured content the sequence can be clearly defined and enforced through automated validation of the standard.

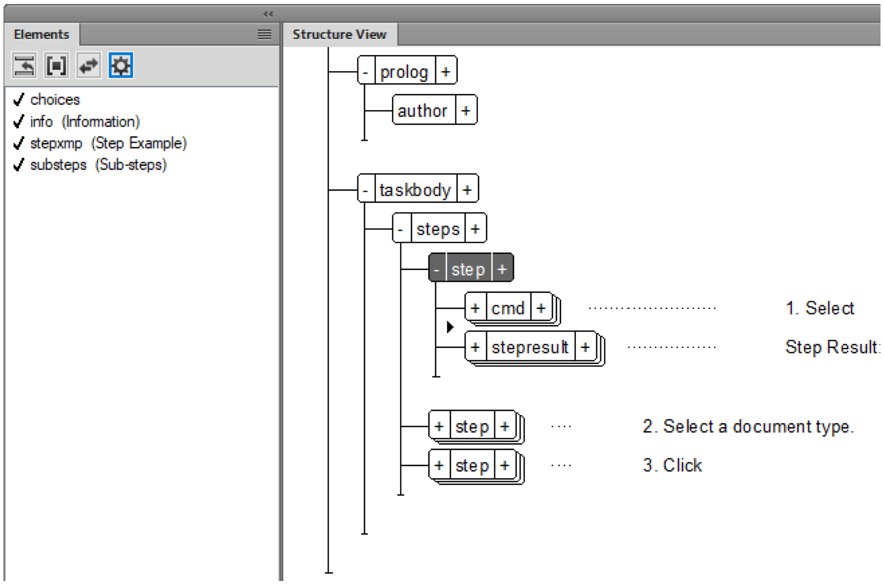
EXAMPLE: A **step** could have a **command** (instruction to perform an action), a **stepxmp** (a step specific example illustrating a specific way to perform a step), a set of **choices**, a set of **substeps**, or a specific **stepresult**.

This establishes a clear and defined guide to creating a <**step**>. It now represents the action a user must follow to complete the single step accurately. A collection of <**step**> elements would be grouped into <**steps**> (note the singular <**step**> and the plural <**steps**> here...) and these would be the core of the task body. Each <**step**> element would require:

- At a minimum, a clear command with a specific action the user would perform
- One or more optional choices to consider, informative statements to help your reader understand the step, a specific example of the step or substeps to perform
- An optional expected result that identifies the outcome of the step (if it's not obvious to the reader)
- An optional way to troubleshoot the step if the result is not as expected

This model ensures that the command is first, then a list of optional "middle" content, and the result is last. In short, the model enforces a standard and consistent organization.

Figure 6: Enforced content order



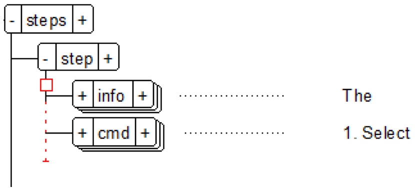
EXAMPLE: Using a sample Structure View (the right portion of **Figure 6**) and clicking between a <**cmd**> and <**stepresult**> shows a small black triangle. It also updates the Elements (left portion of the image) to display only 4 valid options for an author to choose from. The author can still create content, but must do so within the confines of the available structured options.

This could be formalized as follows:

- <step> must begin with a single <cmd>
- The <cmd> may be followed by the following optional and repeatable elements: <choices>, <info>, <stepxmp>, <substeps>
- The <step> may also, optionally, contain a final element listing the <stepresult>

If required content is missing or inserted and reorganized in a structurally invalid order it breaks the rules as seen in Figure 7. At times it may be important to reorganize content and, briefly, have invalid structures. Good software can validate the structure and provide real-time updates to identify any invalid content.

Figure 7: Invalid structure representation



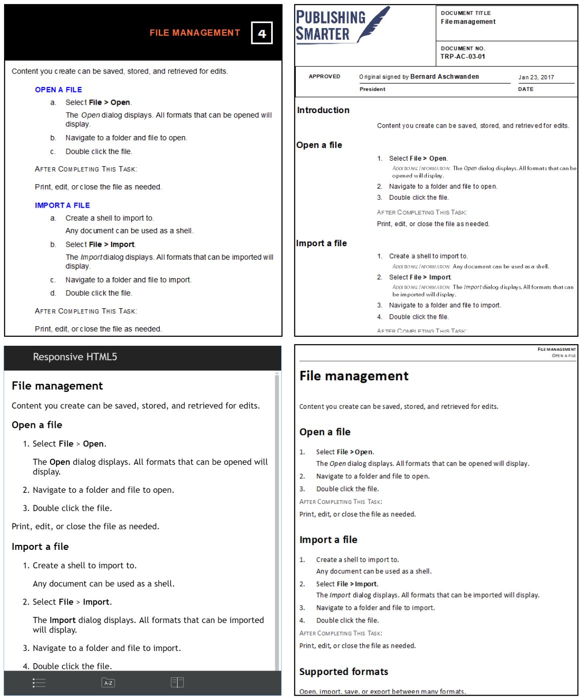
Here the small red square error mark indicates a missing element and the dashed descended line indicates an element in the incorrect location. Based on the formalized rules (above) the step must begin with a single <cmd> and this may be followed (optionally) by the <info>. As this is not the case, the error is flagged.

Standardize corporate branding

As content between traditional silos is starting to be shared more often (such as marketing and technical communications converging[1] or training and policy and procedure sharing step-by-step instructions) it's also important to reformat content based on the specific need of a corporate brand. To facilitate file sharing, structured content can be exchanged between software tools. As long as both tools "understand" the same structured code, they can reformat information dynamically to display as needed.

EXAMPLE: A structured document is opened in multiple tools, and uses multiple templates. The document may have a different appearance when published in a smaller page layout, if published as a work order, included in a mobile app, or when opened using a default author template. Figure 8 shows that while content is identical in every configuration, the appearance of the content dynamically changes and adheres to the corporate template, regardless of what that template looks like. Authors could even have a web-based template to write in that is completely removed from the appearance of any review or publish template. Here the same content appears in unique formats based on tools and templates.

Figure 8: Same content, unique templates



FILE MANAGEMENT 4

Content you create can be saved, stored, and retrieved for edits.

OPEN A FILE

a. Select **File > Open**.

The *Open* dialog displays. All formats that can be opened will display.

b. Navigate to a folder and file to open.

c. Double click the file.

After Completing This Task:

Print, edit, or close the file as needed.

IMPORT A FILE

a. Create a shell to import to.

Any document can be used as a shell.

b. Select **File > Import**.

The *Import* dialog displays. All formats that can be imported will display.

c. Navigate to a folder and file to import.

d. Double click the file.

After Completing This Task:

Print, edit, or close the file as needed.

PUBLISHING SMARTER

DOCUMENT TITLE
File management

DOCUMENT NO.
TRP-AC-03-01

APPROVED Original signed by Bernard Aschwanden — President — Jan 23, 2017 — DATE

Introduction

Content you create can be saved, stored, and retrieved for edits.

Open a file

1. Select **File > Open**.
   Additional Information: The *Open* dialog displays. All formats that can be opened will display.
2. Navigate to a folder and file to open.
3. Double click the file.

After Completing This Task:

Print, edit, or close the file as needed.

Import a file

1. Create a shell to import to.
   Additional Information: Any document can be used as a shell.
2. Select **File > Import**
   Additional Information: The *Import* dialog displays. All formats that can be imported will display.
3. Navigate to a folder and file to import.
4. Double click the file.

After Completing This Task:

Responsive HTML5

File management

Content you create can be saved, stored, and retrieved for edits.

Open a file

1. Select **File** > **Open**.

   The **Open** dialog displays. All formats that can be opened will display.

2. Navigate to a folder and file to open.
3. Double click the file.

Print, edit, or close the file as needed.

Import a file

1. Create a shell to import to.

   Any document can be used as a shell.

2. Select **File** > **Import**.

   The **Import** dialog displays. All formats that can be imported will display.

3. Navigate to a folder and file to import.
4. Double click the file.

File management
Open a file

File management

Content you create can be saved, stored, and retrieved for edits.

Open a file

1. Select **File > Open**.
   The *Open* dialog displays. All formats that can be opened will display.
2. Navigate to a folder and file to open.
3. Double click the file.

After Completing This Task:

Print, edit, or close the file as needed.

Import a file

1. Create a shell to import to.
   Any document can be used as a shell.
2. Select **File > Import**.
   The *Import* dialog displays. All formats that can be imported will display.
3. Navigate to a folder and file to import.
4. Double click the file.

After Completing This Task:

Print, edit, or close the file as needed.

Supported formats

Open, import, save, or export between many formats.

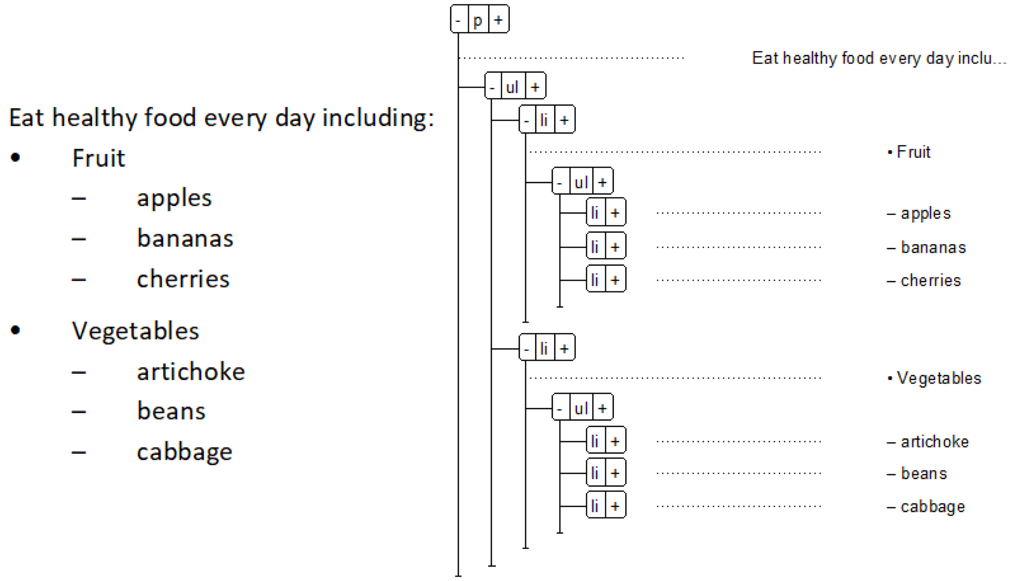
Automating and enforcing format

Structured content removes the need for authors to think about the style-based format assigned to tables, words, paragraphs, or other content. Content is not organized into formatted units based on tagging such as a heading or a subheading. Format no longer drives the relationship of the content. A unit of information can be logically reorganized, and is formatted automatically, based on context of use.

[1] https://offers.adobe.com/en/na/marketing/landings/xml_documentation_add_on_for_adobe_experience_manager_whitepaper_the_convergence_of_technical_and_ma.html

EXAMPLE: An unordered list may appear nested within another list. **Figure 9** shows three lists. One has two bullets; the first for Fruit and the second for Vegetables. The others two lists are specific lists of fruits and of vegetables.

Figure 9: List and nested-list formats

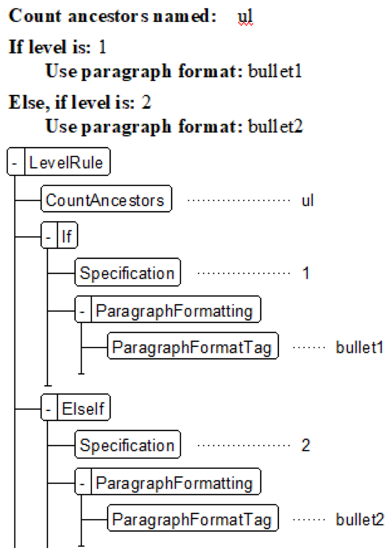


Eat healthy food every day including:

- Fruit
  - – apples
  - – bananas
  - – cherries
- Vegetables
  - – artichoke
  - – beans
  - – cabbage

The format is automatically assigned based on the nested level. If the list is the "first" in the hierarchy it appears as a bullet list with round bullets. If placed at a second level the list is automatically formatted as a dash list, and so on. Should the lists later be reorganized the formats would automatically change to match the rules.

Figure 10: Format rules based on list count



Count ancestors named: ul
If level is: 1
Use paragraph format: bullet1
Else, if level is: 2
Use paragraph format: bullet2

As an author there is no need to reformat list content if it is reorganized or reused. With the right tools list formats are automatically assigned based on the number of nested <**ul**> elements. **Figure 10** shows rules to dynamically format <**li**> elements as a main or second level bullet.

Using metadata for versioning and delivery

Metadata is, as mentioned previously, information about information. This can be used to perform additional actions. Just like an <image> may need to know the actual file to import, the height and width, and other data, you can perform additional actions based on metadata.

Table 4: Procedural steps sample

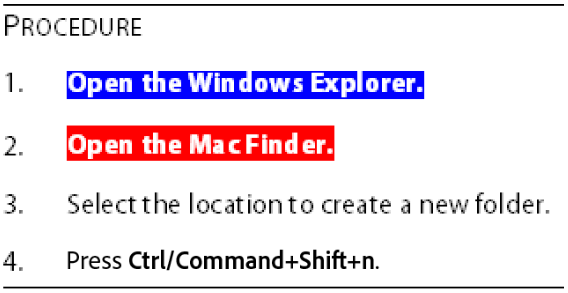
```
<steps>
  <step platform="win">Open the Windows
  Explorer.</step>
  <step platform="mac">Open the Mac
  Finder.</step>
  <step>Select the location to create a new
  folder.</step>
  <step>Press Ctrl/Command+Shift+n.</
  step>
</steps>
```

EXAMPLE: In **Table 4** a set of procedural steps are customized for a specific platform. That is, a step may be unique to either the Windows or the

Macintosh operating system. The step would appear for authors, but conditionally be included in content for specific readers based on the platform. In this sample we've flagged two steps to stand out.

An author would see all the content, perhaps formatted similar to Figure 11. The information is automatically numbered with 4 steps (which to the author are visible and can be edited, proofread, translated, or worked with as needed before it is seen by the consumer of the content).

Figure 11: Content as seen by an author

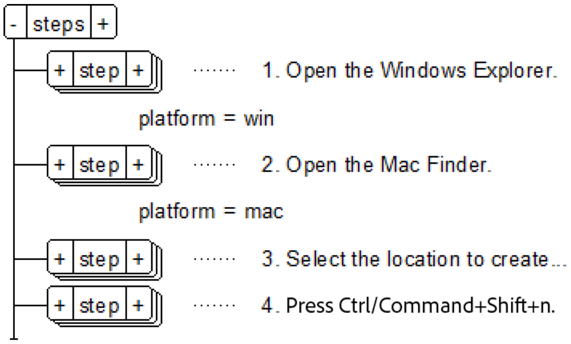


PROCEDURE

1. Open the Windows Explorer.
2. Open the Mac Finder.
3. Select the location to create a new folder.
4. Press **Ctrl/Command+Shift+n**.

The structure of this content has <step> elements. There are optional attributes with specific values assigned to them.

Figure 12: Structure with platform attributes

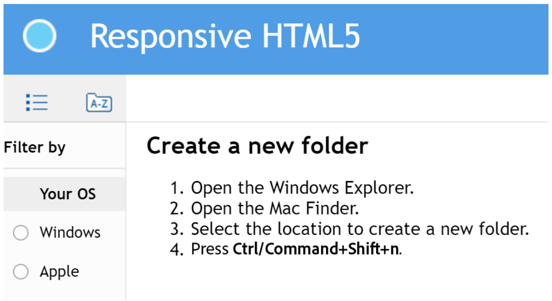


- steps +
+ step + ... 1. Open the Windows Explorer.
platform = win
+ step + ... 2. Open the Mac Finder.
platform = mac
+ step + ... 3. Select the location to create...
+ step + ... 4. Press Ctrl/Command+Shift+n.

The audience sees information specific ONLY to their environment (the reader may even toggle the display to see one or the other in some outputs). For example, the default may show all content, but also has an option to filter by OS.

This enhances value to the reader and adds minimal overhead in the content creation process. It also means a single source of content exists which helps reduce risk, costs, and management.

Figure 13: Responsive HTML5 with all content



Responsive HTML5

Filter by

Your OS

○ Windows

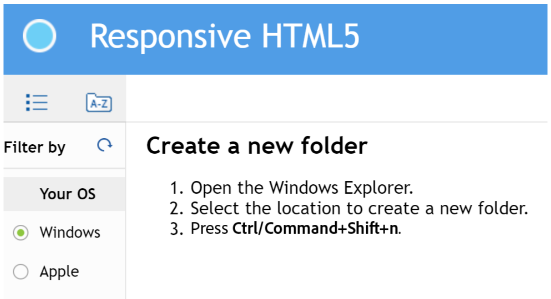
○ Apple

Create a new folder

1. Open the Windows Explorer.
2. Open the Mac Finder.
3. Select the location to create a new folder.
4. Press **Ctrl/Command+Shift+n**.

If the Windows version is shown then only three steps are displayed.

Figure 14: Filtered to show only Windows



Responsive HTML5

Filter by

Your OS

● Windows

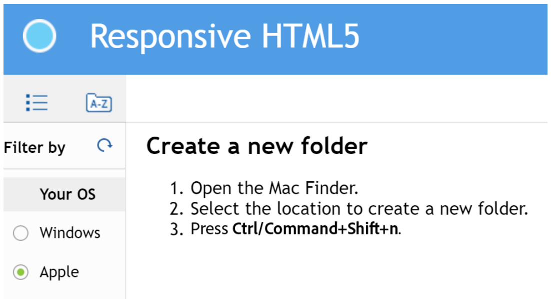
○ Apple

Create a new folder

1. Open the Windows Explorer.
2. Select the location to create a new folder.
3. Press **Ctrl/Command+Shift+n**.

If the Macintosh version is shown then the steps update to show only the Finder procedure.

Figure 15: Filtered to show only Macintosh



Responsive HTML5

Filter by

Your OS

○ Windows

● Apple

Create a new folder

1. Open the Mac Finder.
2. Select the location to create a new folder.
3. Press **Ctrl/Command+Shift+n**.

Ideally, the author also configures step 3 to contain only an OS specific phrase so that only Windows or Macintosh content displays. Even this can be done with effective processes and tools.

Publishing to multiple output formats

Structured content allows output to almost any format. One source can therefore be converted to be read on any device, or to any print ready template. Need to create for tablets, phones, or desktops? No problem. You can even publish to mobile apps! Do you want PDF and print-ready content for newsletters, whitepapers, print books (hardcover or paperback), textbooks, catalogs, or any other "traditional" print deliverable? You can generate it all from one source.

EXAMPLE: A publisher creates structured content explaining how to prepare a floor for installation of ceramic tile. This is used in a Planning Guide (delivered online), referenced in educational materials (print and online), provided to customers in-store during hands-on demonstrations, and even is used online in videos as part of an overall strategy to gain new customers. This means there is a need to publish to many outputs.

Using a traditional model the content is reformatted for each print component. It is converted to HTML content and plugged into each required location with updates to stylesheets and other format elements.

With an XML based workflow we can convert the content to many outputs, and can even do so in an automated way. This frees up writers, editors, managers, and others to do the work they were hired to do, not convert content through labor-intensive manual workflows!

Let's explore these ideas in **Table 5**. Notice that the structured content has steps and substeps. Some steps also have additional information that help guide the reader to purchase product.

Table 5: Structured <steps> and <substeps>

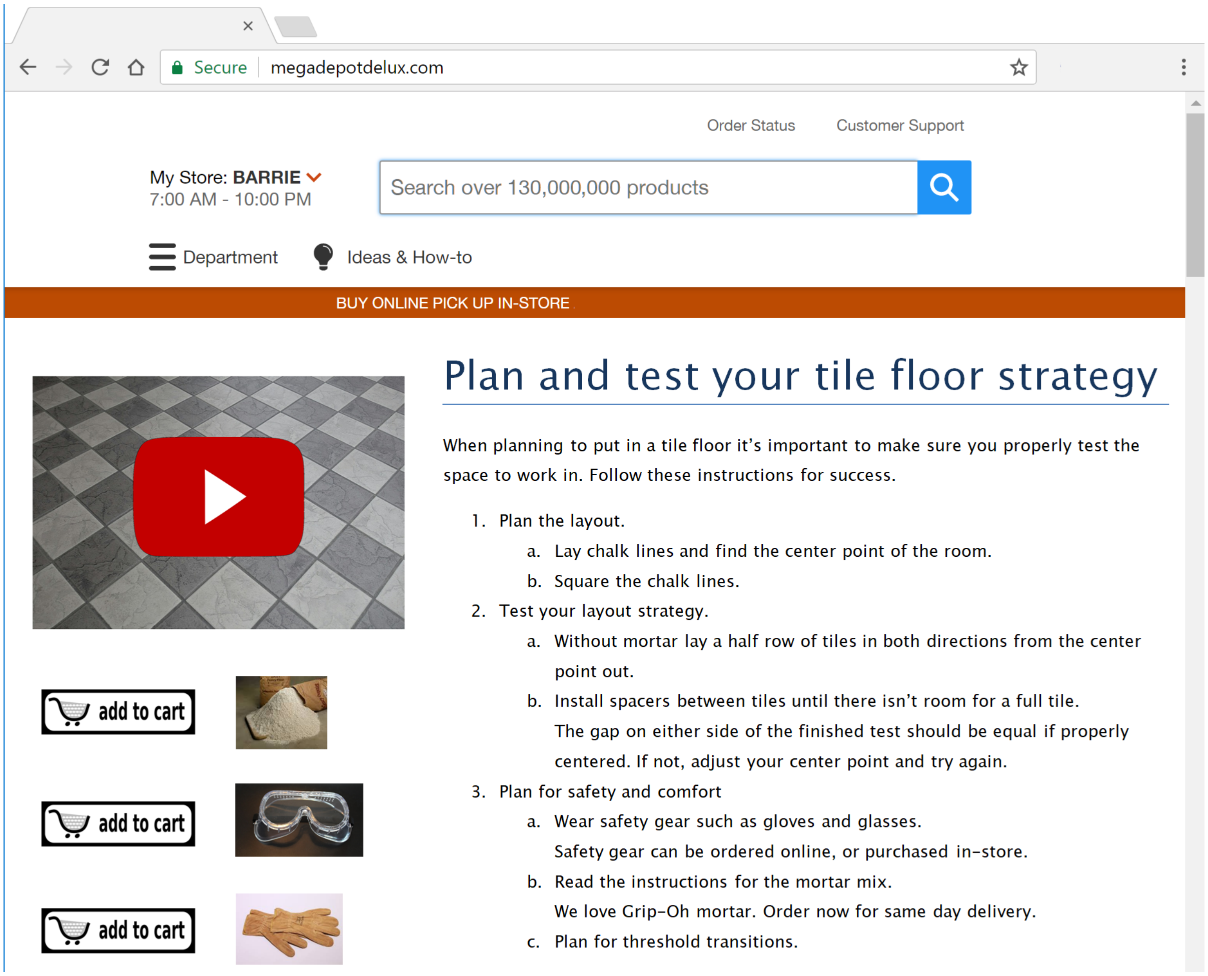
```
<steps>
  <step>Plan the layout.
    <substeps>
      <step>Lay chalk lines and find the center point of the room.</step>
      <step>Square the chalk lines.</step>
    </substeps>
  </step>

  <step>Test your layout strategy.
    <substeps>
      <step>Without mortar lay a half row of tiles in both directions from the center point out.</step>
      <step>Install spacers between tiles until there isn't room for a full tile.
        <info>The gap on either side of the finished test should be equal if properly centered. If not, adjust your center point and try again.</info>
      </step>
    </substeps>
  </step>

  <step>Plan for safety and comfort
    <substeps>
      <step>Wear safety gear such as gloves and glasses.
        <info>Safety gear can be ordered online, or purchased in-store.</info>
      </step>
      <step>Read the instructions for the mortar mix.
        <info>We love Grip-Oh mortar. Order now for same day delivery.</info>
      </step>
      <step>Plan for threshold transitions.</step>
      </substeps>
  </step>
</steps>
```

When delivered, content can be formatted in many ways to instruct someone on planning for tile installation, or as part of a sales discussion, or even online with links to specific product. In this way, the same content is delivered in multiple formats from one source. In this sample we've simplified the code for the purpose of the document, but a lot more information could be stored in the materials and published to specific audiences and outputs.

Structured content is converted using a logical set of instructions.

This information may be used by marketing on the corporate website as text embedded on a page with a video showing the product in use and the option to immediately buy product or related materials. Your content is now used to help generate revenue!

Figure 16: Marketing (Website)



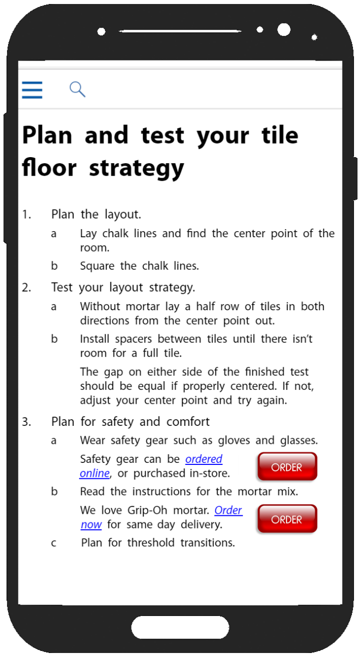
Plan and test your tile floor strategy

When planning to put in a tile floor it's important to make sure you properly test the space to work in. Follow these instructions for success.

1. Plan the layout.
    a. Lay chalk lines and find the center point of the room.
    b. Square the chalk lines.
2. Test your layout strategy.
    a. Without mortar lay a half row of tiles in both directions from the center point out.
    b. Install spacers between tiles until there isn't room for a full tile.
    The gap on either side of the finished test should be equal if properly centered. If not, adjust your center point and try again.
3. Plan for safety and comfort
    a. Wear safety gear such as gloves and glasses.
    Safety gear can be ordered online, or purchased in-store.
    b. Read the instructions for the mortar mix.
    We love Grip-Oh mortar. Order now for same day delivery.
    c. Plan for threshold transitions.

Now think about how it can be used as part of a mobile application. Not only are the detailed step-by-step instructions provided, but so in the option to order materials immediately. Figure 17 show that, again, the content can be used as a sales generation tool to help build revenue.

Figure 17: Mobile (for DIY)

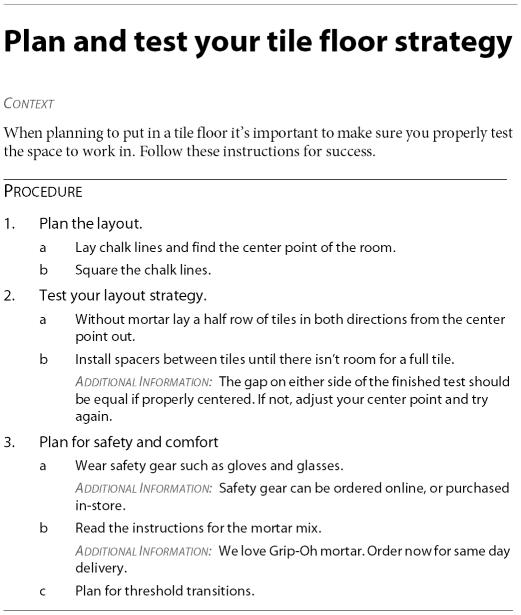


Plan and test your tile floor strategy

1. Plan the layout.
   a Lay chalk lines and find the center point of the room.
   b Square the chalk lines.
2. Test your layout strategy.
   a Without mortar lay a half row of tiles in both directions from the center point out.
   b Install spacers between tiles until there isn't room for a full tile.
   The gap on either side of the finished test should be equal if properly centered. If not, adjust your center point and try again.
3. Plan for safety and comfort
   a Wear safety gear such as gloves and glasses.
   Safety gear can be ordered online, or purchased in-store. ORDER
   b Read the instructions for the mortar mix.
   We love Grip-Oh mortar. Order now for same day delivery. ORDER
   c Plan for threshold transitions.

The content can also be added to a printed or PDF brochure where the primary steps are all we need. Figure 18 should look familiar as it represents what we may see in a manual or as a single page instruction on product use.

Figure 18: Brochure (Print)



Plan and test your tile floor strategy

Context

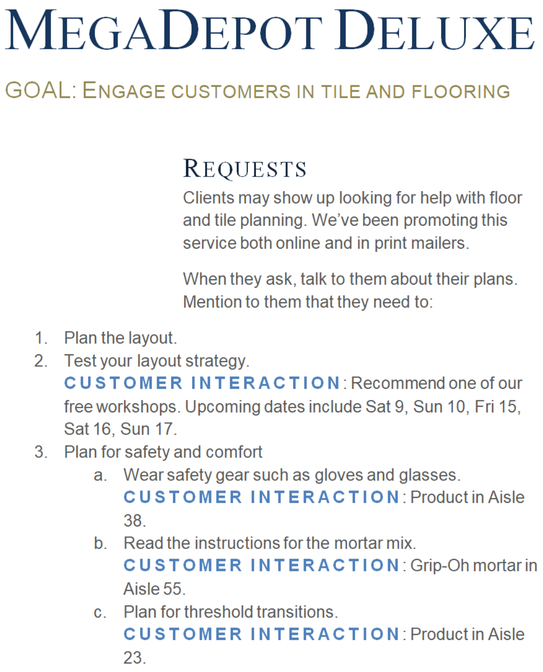
When planning to put in a tile floor it's important to make sure you properly test the space to work in. Follow these instructions for success.

Procedure

1. Plan the layout.
   a Lay chalk lines and find the center point of the room.
   b Square the chalk lines.
2. Test your layout strategy.
   a Without mortar lay a half row of tiles in both directions from the center point out.
   b Install spacers between tiles until there isn't room for a full tile.
   Additional Information: The gap on either side of the finished test should be equal if properly centered. If not, adjust your center point and try again.
3. Plan for safety and comfort
   a Wear safety gear such as gloves and glasses.
   Additional Information: Safety gear can be ordered online, or purchased in-store.
   b Read the instructions for the mortar mix.
   Additional Information: We love Grip-Oh mortar. Order now for same day delivery.
   c Plan for threshold transitions.

Finally, let's explore what an associate in a store needs. Figure 19 has additional information (normally hidden from any consumer) included as part of the content. Associates can connect content to client engagement and even consider specific interactions to help generate revenue.

Figure 19: Store Associates Only (Print)



MegaDepot Deluxe

Goal: Engage customers in tile and flooring

Requests

Clients may show up looking for help with floor and tile planning. We've been promoting this service both online and in print mailers.

When they ask, talk to them about their plans. Mention to them that they need to:

1. Plan the layout.
2. Test your layout strategy.
   CUSTOMER INTERACTION: Recommend one of our free workshops. Upcoming dates include Sat 9, Sun 10, Fri 15, Sat 16, Sun 17.
3. Plan for safety and comfort
   a. Wear safety gear such as gloves and glasses.
      CUSTOMER INTERACTION: Product in Aisle 38.
   b. Read the instructions for the mortar mix.
      CUSTOMER INTERACTION: Grip-Oh mortar in Aisle 55.
   c. Plan for threshold transitions.
      CUSTOMER INTERACTION: Product in Aisle 23.

We can create all these outputs from one source.

Each contains a mix of content customized based on the delivery and output. On the web it is merged with video and instructional content. On a mobile device a reader can immediately order the products discussed. For a brochure it's a simple black-and-white document. Lastly, in-store customers are engaged through additional information about workshops and specific aisles with products (this also helps to drive sales!).

Each format has a unique look, and a specific function, but all of them are based on one single source of content.

Supporting content reuse

The world of XML and DITA has opened up content reuse beyond what you may traditionally do with an authoring tool. If you are familiar with concepts such as text insets, variables, conditional content, and sharing files between books, you are already on the way to working with structured content reuse.

Many mechanisms exist for content reuse. Some of the easier ways to explain and demonstrate them connect to what you already know about your authoring tool. As discussed earlier, there are ways to assign metadata such as the platform (we compared the same general steps for both Windows and Macintosh), or to create custom output based on templates. However, within DITA there are additional reuse components.

The <**ph**> element defines a phrase. In the DITA specification the <**ph**> is specifically created for reuse or conditional processing (for example, when specific parts of a paragraph only apply to a given audience, platform, or product).

Share content with others using XML

Sharing content has become a huge topic of discussion in the technical communications world, but it's also a big deal to people in marketing, sales, training, and basically any other place people use and share content. It's now simpler to create XML (specifically, DITA) as the standards have developed and fewer companies are "building their own". With the implementation of DITA as a standard you can also create and share with people using other tools.

EXAMPLE: XML lets you move information from one authoring tool to another without losing information. You can develop information in one tool, save it out to XML, and then open the XML files in another tool. Perhaps a technical writing group uses FrameMaker and needs to share information with a marketing group who use Adobe Experience Manager Guides. You can use XML to exchange content between two or more applications with ease.

Reducing localization cost

Part of the cost of translation is related to reformatting content that is translated. Page durations will change, images may be replaced with larger or smaller ones, tables may reflow, or other updates may be required based on the revised materials. Exact numbers will vary for specific projects, but as a general rule, about half the total localization cost is for publishing and production. A publishing workflow built on XML and structured authoring lets you automate much of the publishing effort, so you can greatly reduce the ongoing costs associated with localization.

Simplifying database publishing

Databases can generate an XML file with all the information needed for publishing. This allows you to export content from the database and then import it into a publishing tool for output to PDF or HTML based deliverables.

EXAMPLE: A database of parts includes the part ID, name, illustration, pricing (both retail and wholesale), and other related information. Specific content is exported every month as XML content. This content is imported to a publishing template. All layout and format is automatically applied. The result is a high quality PDF document used by sales reps at trade shows, sent to clients, and provided to partners. The same content is also published to HTML5 output and converted to the most current version of the company app.

Complying with required document structures

Regulatory environments may specify specific content that is required in a specific order. Structured authoring allows consistent content development to ensure compliance with government or industry requirements.

Next steps

Let's explore how software can best be used to help you migrate content to a structured format. In **Section 2: Benefits of structured FrameMaker** we'll explore Adobe FrameMaker and its features. Specifically, we'll use Adobe FrameMaker (2019 release) and demo features that are available even in just the trial version. If you don't have FrameMaker, don't worry. We'll provide links to videos that demonstrate ideas so you can follow along, as well as tutorials and access to the trial version to get you started. If you already have FrameMaker then it's even easier!

Section 2: Benefits of structured FrameMaker

XML can be complex and may not always look like you would prefer. FrameMaker uses an author-centric structured environment that allows you to create valid XML content. You can create, edit, and publish complex content with rich format and layout options. Authors can create PDF and online formats with all the same functions they expect from their software. Structured FrameMaker content gives you the comfort of an authoring tool with a powerful visual interface, and lets you enjoy the benefits of a structured workflow.

- **Migrating from unstructured FrameMaker to structured FrameMaker**
- **Authoring visually**
- **Creating multiple outputs from one source**
- **Relatively low licensing cost**
- **Relatively low implementation effort**
- **XML implementation options**

Migrating from unstructured FrameMaker to structured FrameMaker

Traditional unstructured content is based on paragraph, character, table, and other formats that are stored in catalogs. Content in a structured workflow is managed based on the context of use and format automatically applied. That is, a paragraph that might be in a bullet list can be moved to a numbered list and automatically reformatted. Or a topic that is nested 2 levels deep with related indents could be moved in its entirety up 1 level and all subordinate content immediately reformats as well!

While the author experience in a structured environment changes, the capabilities of the software remain the same. Authors can often be up and running in a completely new workflow within days; much faster than they might be if switching between tools completely. Additionally, templates can keep the look and feel they had before the migration, ensuring content retains the look and feel that authors expect.

As a bonus, authors will discover that they no longer need to even think about content format. Instead, they have a consistent, repeatable, and automated format-based document. The formatting rules informally applied to unstructured content can be formalized in a structured workflow.

Authoring visually

There are multiple ways that a structured document can be viewed. This includes with markup (tags) showing in-line with text, markup without tags, or the Structure View with a tree-like hierarchy of elements. FrameMaker supports many combinations, allowing authors to see what they want, when they want.

Creating multiple outputs from one source

Being able to deliver any type of output is a major strength of working with FrameMaker, and structured content makes this even better. While numerous formats can be created out-of-the-box let's talk about the most common formats used today—PDF and HTML5.

Print and PDF output

Just like unstructured FrameMaker, a structured FrameMaker environment creates the best PDF in the world. Most DITA-aware authoring tools either do not have a print option, or they create a finished PDF with no way to customize and configure the format unless you put time into learning programming. Layout for print can be designed in almost the exact same way as with unstructured content. The resulting template is connected to FrameMaker's *Element Definition Document* (EDD) and allows complex print publishing with dependable and repeatable output.

HTML5 output

In the same way that Adobe allows you to quickly convert content to PDF, the ability to create HTML5 content is also native to FrameMaker. The "Publish" option allows you to convert topics or maps to HTML5 including dynamic content filtering, and automated resizing of content based on device type, and does it all with a completely visual interface for customization.

Relatively low licensing cost

Support for structured authoring is already included if you are using FrameMaker. It's simply a toggle in the Preferences. If you want to try it out-of-the-box it's easy to do and you can work with structure without making a huge commitment to new software. This can be scaled in an organization by adding either other licenses of FrameMaker, or even through the use of a *component content management system* (CCMS) like Adobe Experience Manager Guides that allows reviewers, subject matter experts, or others to contribute using only their browser!

Relatively low implementation effort

You already have an unstructured template. The existing paragraph, character, table, cross-reference, and other formats can be referenced and reused. There are also tools built right into FrameMaker to help you convert your content. We'll even teach you how to get started for free!

XML implementation options

When working with FrameMaker and XML you can work with one or more workflows. Not everyone wants to use structured content at every step of the process. Some people want to use FrameMaker as a publishing tool. Others want to integrate FrameMaker with a component content management system. Whatever workflow you want to use, FrameMaker can support it.

- Author structured content in FrameMaker, and as needed you save content as XML. Structured files are saved as FrameMaker documents, opened, edited, and when you need to convert them to XML you "Save As" and generate valid structured text files.
- Open XML files, edit, and export the content back to XML, all from within FrameMaker. FrameMaker is your authoring tool, completely configured so that you open content and see a fully formatted rendition on screen. When you edit, you do so using the complete visual interface with headings, tables, character formats, and other familiar content. When you save files at the end of the day they are saved back to their native XML format.
- Work with a mix of authoring tools, including FrameMaker, to create, edit, and publish content. When it is time to publish, you open the files in FrameMaker and combine all the content to produce the final deliverable. In this environment, contributors could work with online tools or other low-end XML editors to contribute to a FrameMaker based team.
- Database publishing workflows export a database to a structured format that is then imported to FrameMaker for publishing to any supported formats. This allows you to create and deliver content that is based on the most current database and then import that content

to a full page layout and design tool creating a finished document with rich and visually appealing design and all the most current technical specifications and data.

- Author in any tool and publish with FrameMaker. Create content in any tool or system that allows export to XML. Then receive the content, load it into FrameMaker, and produce output regardless of the source tools and processes used.

Section 3: Working in a supported structured authoring environment

Unstructured FrameMaker just needs a high quality template and you can start to author. You define paragraph, character, page, table, and other properties, test the template, train authors, and let them write. However, in a structured authoring environment additional files are required.

- **Authoring only in structured FrameMaker**
- **Authoring in structure (with XML support)**
- **Analyzing content**
- **Choosing an EDD strategy**
- **Get familiar with EDD and XML syntax**

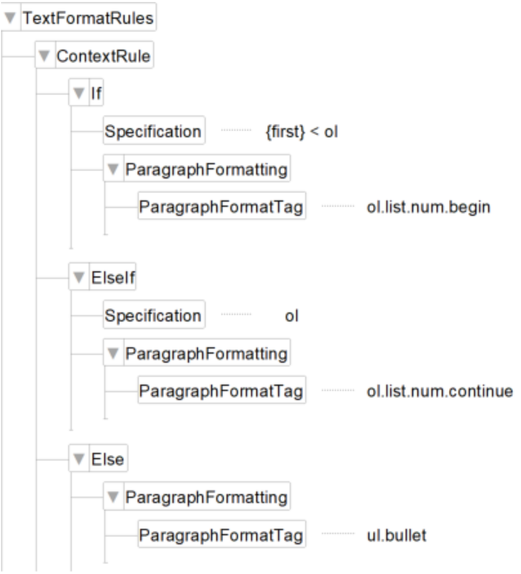
Authoring only in structured FrameMaker

If you don't plan to import or export XML you can still use structured FrameMaker. This provides a formal guided authoring environment, dynamic formatting, and other benefits without the need to immediately work with XML. It's a good way to get started with structured authoring and develop standards beyond a company style guide.

A basic structured authoring environment requires both an EDD and an associated template. With these two documents, rules can be defined and connected to FrameMaker's formatting functions. These methods can even be combined together based on the complexity of your format needs.

- *Connect to a template* with formats. **Figure 20** demonstrates an EDD rule that states "if the item is the first in a numbered list use the paragraph tag ol.list.num.begin, but if it is not the first item, then use the paragraph tag ol.list.num.continue, and in any other situation use a ul.bullet". This allows your structure to connect to a template using tags for paragraphs, characters, tables, and so on, all based on what you already may know about unstructured templates and numbering.

Figure 20: Connecting to paragraph tags



Element (Container): li
General rule: <TEXT>

Text format rules

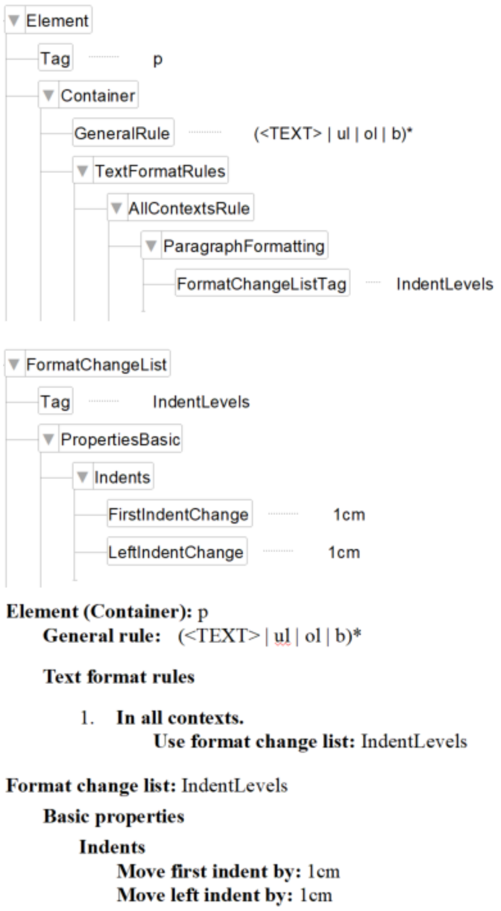
1. **If context is:** {first} < ol
Use paragraph format: ol.list.num.begin

Else, if context is: ol
Use paragraph format: ol.list.num.continue

Else
Use paragraph format: ul.bullet

- *Use format change lists (FCL)*. In **Figure 21** an FCL uses a format that is defined in the EDD, but in one place. For example a format change list may define a level of indent for all secondary level to add a fixed indent for first line and for left line in a paragraph, but use default paragraph settings for all other elements. This allows you to, for example, initially decide every nested topic is indented with an extra 3cm of space. Then later, you make one change and update this to 2cm of space globally. Done. Everything updates. No more tweaking 15 paragraph tags one at a time every time someone redefines an indent.

Figure 21: Format change list references



Element
Tag p
Container
GeneralRule (<TEXT> | ul | ol | b)*
TextFormatRules
AllContextsRule
ParagraphFormatting
FormatChangeListTag IndentLevels

FormatChangeList
Tag IndentLevels
PropertiesBasic
Indents
FirstIndentChange 1cm
LeftIndentChange 1cm

Element (Container): p
General rule: (<TEXT> | ul | ol | b)*

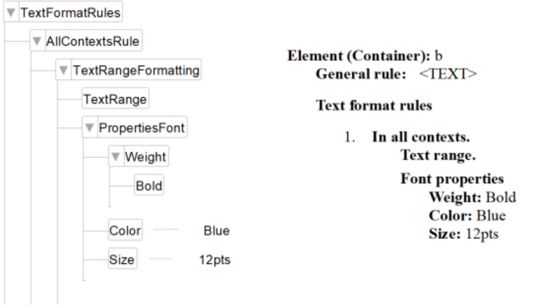
Text format rules

1. **In all contexts.**
Use format change list: IndentLevels

Format change list: IndentLevels
Basic properties
Indents
Move first indent by: 1cm
Move left indent by: 1cm

- *Embed formatting in each element.* Figure 22 shows a format rule in the actual element and ensures the exact value is applied to only the one element in all specific use cases. This ensures that, if you decide an element should be bold, blue, and 12pt in height it is ALWAYS that specific value, regardless of other factors.

Figure 22: Embedded format rule



TextFormatRules
AllContextsRule
TextRangeFormatting
TextRange
PropertiesFont
Weight
Bold
Color Blue
Size 12pts

Element (Container): b
General rule: <TEXT>

Text format rules

1. **In all contexts.**
Text range.
Font properties
Weight: Bold
Color: Blue
Size: 12pts

In all of these examples you need a basic template to define master pages, headers and footers, automated page layout, and other publishing features. How format is defined and applied can be controlled using ideas discussed here.

Once an EDD is created and formatting is decided on combine these ideas into one structured template. To do so you create an EDD and a format-rich template. With both open import the EDD into the initially unstructured template to create a new structured template. Test is and resolve bugs or issues and distribute the finished to authors. Keep the original EDD and a copy of the template in case you ever need updates.

Authoring in structure (with XML support)

There are a few core requirements for structured authoring. Authors use use a structured template and are ready to go. Structured template developers need, at an absolute minimum, an EDD and a template. Ideally both are interconnected. That is, the template has the look and feel you want, and the EDD has the structural rules.

There may be a few additional files needed to get a complete XML-based workflow that allows you to open XML files, edit them, and then save them back into their source XML structure.

Figure 23: XML and FrameMaker workflow

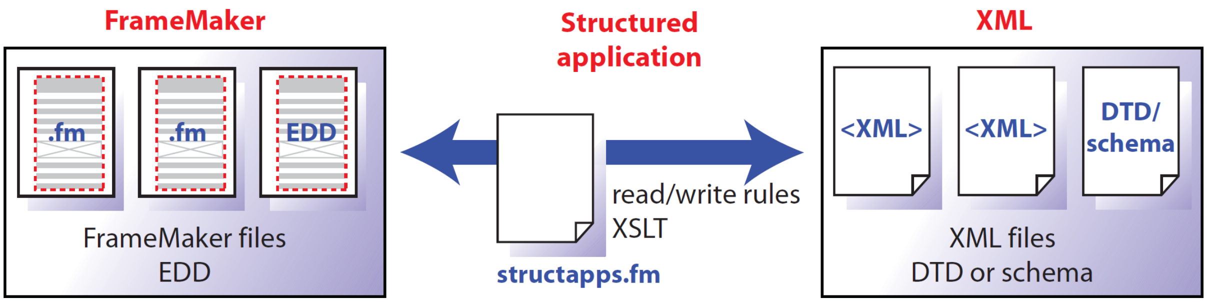


Figure 23 illustrates how content is converted into and out of FrameMaker and XML. Many documents may be required to provide core support to import or export content, and these include:

- **Read/Write Rules** (required): Rules that convert XML content to and from FrameMaker content (for example, XML markup sees a paragraph or an index as a string of code [<p> for paragraph, or <index> for an index entry], but must convert them to FrameMaker text or markers) as you open/save/close an XML file.
- **XSLT** (optional): As XML files are read into FrameMaker (or FrameMaker files exported to XML) there may be additional functions that read/write rules cannot support. If so, code can transform content and even re-organize it.

To author content, assign format, or perform other work within a structured environment, additional required or optional files are developed. These include:

- **Schema** (optional) or **DTD** (required): An XML based set of structural rules that match the EDD and validates XML content.
- **EDD** (required): Structure rules and formats.
- **Template** (required): Import the EDD to the template and format rules are connected to content in the FrameMaker model (such as paragraph/character/table styles, cross-reference formats, page layout, etc.).
- **FrameMaker API Clients** (optional): More sup-port functions can be programmed and refer-enced. For example, if using Adobe Experience Manager Guides you may opt to use an API to connect your XML directly to the repository.
- **Structured Application** (required): This file contains a collection of paths and information about all files used to convert content to and from FrameMaker and XML. This specifies what FrameMaker files and EDD to use, scripts to run, read/write rules to use, DTD to reference, and other setting that, once configured, apply behind the scenes to make import and export of content simple.

While it may sound complex at first the overall workflow could be broken into two major environments for major roles as seen in Figure 24.

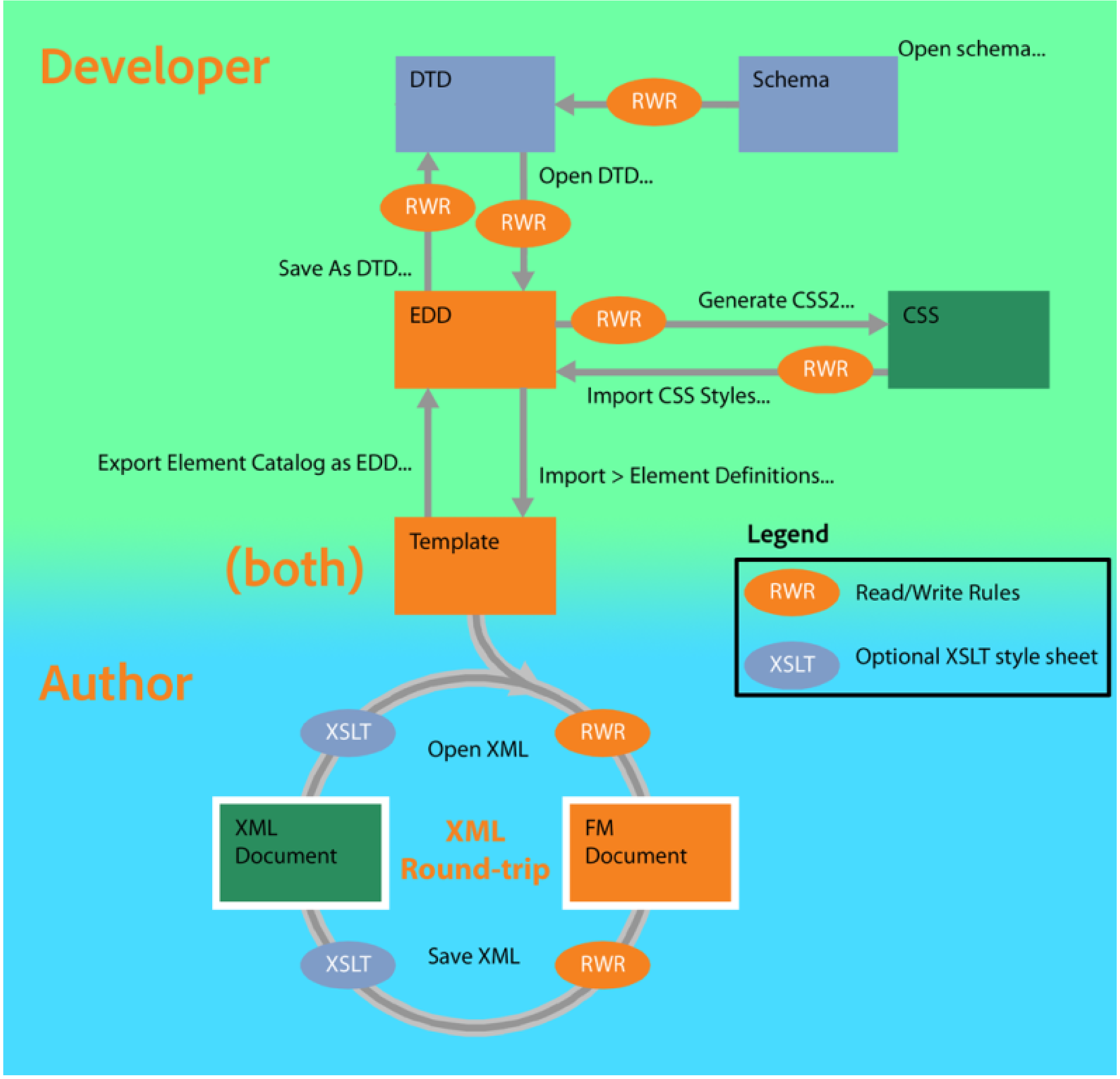
- **Developer**: Creates and manages the template, largely through interaction with the EDD. The developer imports a schema or DTD into FrameMaker and creates an EDD. The EDD contains rules and formats that are imported to the template. Once tested and validated, the template is shared with an author.
- **Author**: Creates, edits, or publishes content, largely by using the template. The template is used to display XML content and formats the content in a human-readable and nicely formatted way.

Both developers and authors work with a shared template, but in two different ways.

- Developers create the template and define the appearance of it
- Authors create content with it.

It is entirely possible (even likely) that the template developer is also an author who uses it.

Figure 24: Structured content workflow roles



Analyzing content

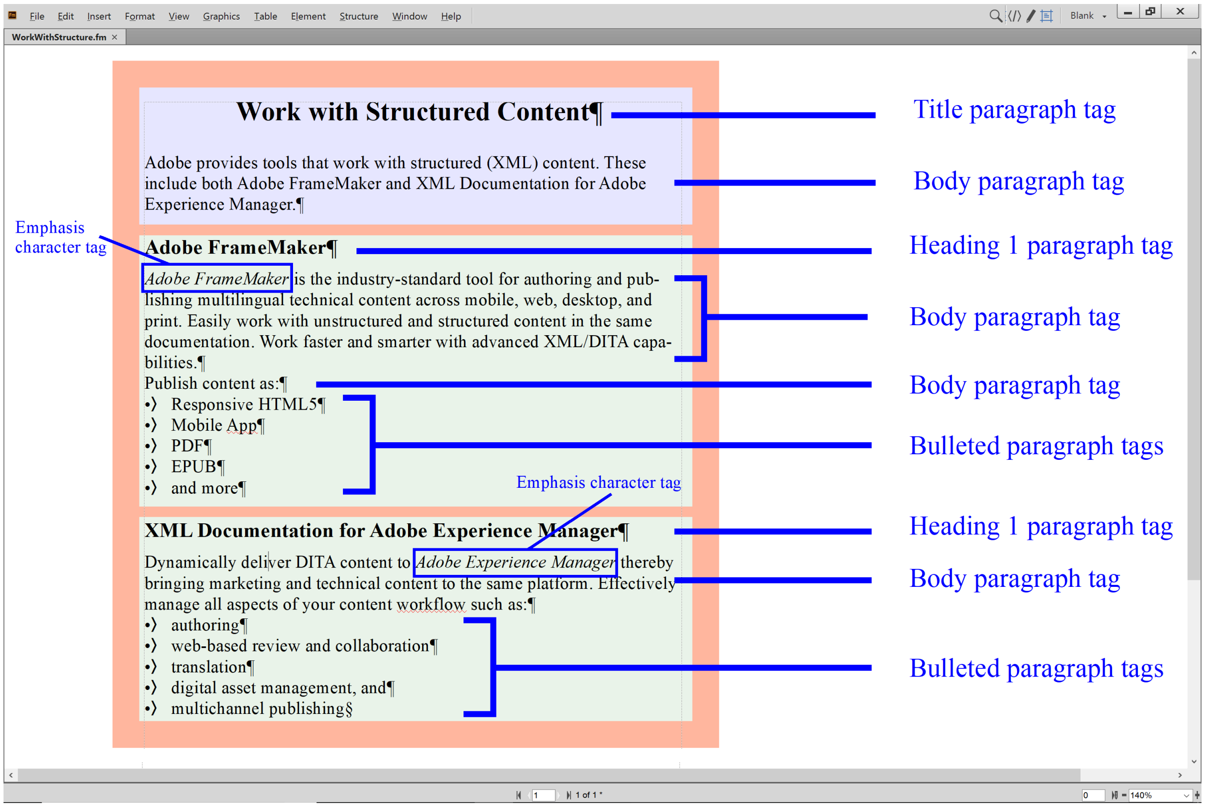
You can get started with structure right out-of-the-box with FrameMaker. However, to understand the core components of a structured application it may be helpful to build your own setup. While this can be complex, based on the content you have, in the following section we'll create a sample document, all the tools required to convert it to structure, and provide you a way to gain insight to the process.

Consider Figure 25 in which there is an analysis of a topic. In it there is an introduction to an idea, sample content in paragraphs, lists, and some semantic character level markup.

Later, in the hands-on exercises, we'll work to build an entire working environment that supports this structured approach to content. We'll reuse this basic analysis to develop all the moving parts needed to migrate the content.

Reviewing the content you can identify the overall topic, a title, several sections, and many paragraphs. There are also lists and character level formats.

Figure 25: Sample topic with tagging



Work with Structured Content¶

Adobe provides tools that work with structured (XML) content. These include both Adobe FrameMaker and XML Documentation for Adobe Experience Manager.¶

Adobe FrameMaker¶

Adobe FrameMaker is the industry-standard tool for authoring and publishing multilingual technical content across mobile, web, desktop, and print. Easily work with unstructured and structured content in the same documentation. Work faster and smarter with advanced XML/DITA capabilities.¶

Publish content as:¶

•) Responsive HTML5¶
•) Mobile App¶
•) PDF¶
•) EPUB¶
•) and more¶

XML Documentation for Adobe Experience Manager¶

Dynamically deliver DITA content to *Adobe Experience Manager* thereby bringing marketing and technical content to the same platform. Effectively manage all aspects of your content workflow such as:¶

•) authoring¶
•) web-based review and collaboration¶
•) translation¶
•) digital asset management, and¶
•) multichannel publishing§

Choosing an EDD strategy

Building an EDD can be based on several workflows. In most cases you end up working with many of the same ideas, but how you get started depends a bit on your goals. In all cases you end up having a set of content rules attached to format rules. The good news is that a lot of the work required in any EDD work has overlapping ideas and rules. It is a bit like technical communications; once you know core grammar rules, style use, and software to work with it won't matter much if it's a user guide, reference manual, or a training video you need to create—the rules you follow are largely the same!

In our sample document there are minimal formats, so the work to build a template is simple. Again, later in this document we'll walk you through a step-by-step sample.

Build your own

To build your own EDD means knowing the end goal of your structure. This can be very simple (imagine a structure for a business memo) or can become incredibly complex (imagine a structure for the rules of setup, maintenance, and operation of a complex medical device, computer system, or airplane).

Building your own is most likely required when you have already evaluated structures that exist and decided they don't meet your needs. You then need to have a complete architecture of your content (either already defined, or bring in a content strategist with structured information architecture experience) and the skills to apply this to your new structure.

We've taken this approach with clients; it works well, but it is usually more complex and costly unless we're looking at a basic document structure. This was popular 20 years ago when XML was in its infancy, but today you are usually best off to first evaluate existing structures. Building your own isn't a bad idea; it's just not the first approach to take as you get started with XML.

Import an existing DTD or schema

If you are considering an existing structure that is not supported in some way by your current FrameMaker installation you can import a set of rules and then attach formatting. You still need to think about content and the rules for it, and decide what parts of the new rules you want to use though. In many cases even an industry "standard" set of rules (for example, air transportation standards, regulatory medical standards, or even DITA) may contain far more elements and structure rules than you need.

You should likely talk with a content strategist who has structured information architecture experience before making decisions to customize any existing rules. This needs to be done in a way that restricts the potential for a mismatch with others who may be using the entire specification.

We've also worked with existing rules with some clients. It saves them the upfront costs of planning a full structure, is often a bit quicker, and can still be formatted to their specific requirements. It's most commonly done when a standard exists, but is not integrated directly into FrameMaker out-of-the-box.

Modify an existing EDD

This is very similar to importing an existing DTD or schema in scope except that the rules have already been imported. You still need to think about the rules, decide which ones apply to your content, how they apply, and why they do or do not apply.

Customizing an EDD means you must also know the end state of your structure. As the modification is to an existing set of rules it can be simpler, once the rules of the existing structure and format are understood.

The clients we work with that are looking to modify an existing EDD are often either updating and customizing an older structure, or working with DITA as it appears out-of-the-box and then customizing templates and rules.

Work with DITA

Since FrameMaker already supports DITA it's easy to get started with this standard. The rules exist, there are many books written about them, training and online support is simpler to find, and numerous conferences discuss the ideals of a DITA-based workflow—sharing tips and tricks for the same.

EDD customization is simpler when you already know your own templates, but it's also relatively quick to build your own look and feel for DITA with FrameMaker. The visual interface makes it far simpler than writing code and building stylesheets using the DITA Open Toolkit. Once you know the FrameMaker EDD rules and the default templates, you'll start to build your own formatting in minutes. This approach to DITA is more and more common as it becomes the most popular standard for content created by structurally-driven technical communications teams.

Publishing Smarter clients are asking about DITA more and more often. Surveys by Adobe and others support this. DITA is an XML standard users can find a lot of information about and learn with general ease due to the broad support available. Once ready to create content they build maps, topics, and use FrameMaker to publish. Of course, as with any new system, there is an initial learning curve to master, but within days they can start creating content and designing templates. Without FrameMaker they can take days or weeks to get settled into a comfortable writing routine, with some finding that building multiple output types can take weeks or even months of effort!

Get familiar with EDD and XML syntax

There is some very specific syntax that matters when developing the FrameMaker EDD—and it's also used in an XML based DTD (the DTD is a *document type definition* that defines content frequency and order). Once you understand a few basic terms you can start to define rules in an EDD (or read rules in a DTD).

Order rules

Table 6 shows a sample of symbols used to define the order of elements in an EDD (or DTD).

- The comma (,) indicates a specific order
- The pipe (|) indicates an either/or
- Parenthesis (()) indicate logical groupings of elements

Table 6: Order rules

Order	Notes
A, B	All elements in order. The first is A, the second is B. Always appears as AB.
A | B	Only one element. Always appears as either A or as B, but not AB or BA.
A | B, C	Incorrect syntax due to mixed connectors. Cannot be used.
A | (B,C)	Either only A, or both B and C in order. Appears as A or BC, but not AB nor AC.
(A | B), C	Either element A or B, followed by C. Appears as either AC or BC.

Based on this, the following samples can be created (the notes use a friendly term for the elements):

Syntax	Notes
para, list	In order, a paragraph followed by a list.
para | list	One or the other of a paragraph or a list, but never both.
title, para, list	In order, three specific elements. A title, then a paragraph, and then a list.
title, (para | list)	A title which is followed by either a paragraph or a list.
title | (para, list)	Either only a title and nothing else, or just a paragraph followed by a list.
(title | para), list	One option of a title or a paragraph. Once inserted this is followed by a list.

A more robust structure can be created by adding frequency rules to the structural order rules.

Frequency rules

Table 7 shows symbols used to define the frequency of elements in an EDD (or DTD).

- No symbol indicates only one element
- A question mark (?) indicates zero or one
- An asterisk (*) is used to indicate zero or more
- The plus sign (+) indicates one or more

Table 7: Frequency rules

Frequency	Notes
A, B	All elements in order and appearing once per element. Always appears as AB.
A, B?	Element A must appear once. Element B is optional and appears zero or one times. Always appears as either A or AB.
A, B*	Element A must appear once. Element B is optional and appears zero or more times. Appears as at least A, and could be AB, ABB, ABBBBBB (unlimited optional B elements).

Based on this, the following samples can be created (the notes use a friendly term for the elements):

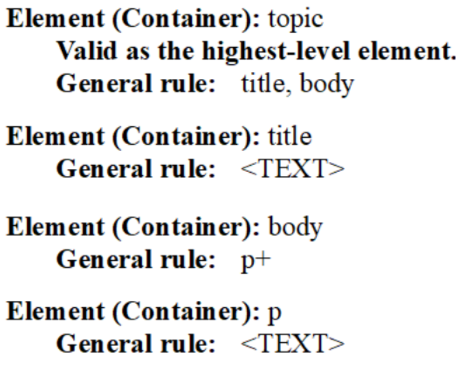
Syntax	Notes
title, para+	One title, followed by as many paragraphs as desired.
title, (para | list)+	One title, followed by a repeatable choice of a paragraph or a list. This could allow a title to be followed by a list, another list, and a third list.
title, (para, list?)+	A title, and at least 1 paragraph. Each paragraph can be followed by an optional list. This ensures that every list has at least one paragraph that leads into it.
title, body	One title, followed by the body. Both elements are required. The body could be further defined to contain a broad range of other elements.

A much more robust structure can be created by adding frequency rules to the structural order rules and creating entire content rule sets.

Sample topic

A very basic topic (simplified for this example) could be defined as seen in Figure 26.

Figure 26: Basic topic sample



Element (Container): topic
  Valid as the highest-level element.
  General rule: title, body

Element (Container): title
  General rule: <TEXT>

Element (Container): body
  General rule: p+

Element (Container): p
  General rule: <TEXT>

That is, a topic contains, in order, a required title element and a required body element:

title, body

The element title contains a rule that specifies a writer creates text:

<TEXT>

The element body contain a rule that one or more paragraphs are required:

p+

Lastly, the <p> element is also defined to contain text and appears as:

<TEXT>

Content created from this could look like Table 8.

Table 8: Basic topic sample content

```
<topic id="fm-structured">
  <title>Work with Structured FrameMaker</title>
  <body>
    <para>Adobe provides tools that work with structured (XML) content. These include both
    Adobe FrameMaker and Adobe Experience Manager Guides.</para>
    <para>Adobe FrameMaker is the industry-standard tool for authoring and publishing multilin-
    gual technical content across mobile, web, desktop, and print…</para>
    <para>Dynamically deliver DITA content to Adobe Experience Manager thereby bringing mar-
    keting and technical content to the same platform.</para>
  </body>
</topic>
```

Let's go further and work hands-on to create an EDD based on this sample. This gives you a chance to get familiar with how an EDD is created, format is assigned, and how to test this.

Section 4: Hands-on development

If you've read everything to this point you should have a good idea of the overall processes to follow. If you skipped it we'll try to make sure that the hands-on information makes sense. Feel free to clarify ideas with our YouTube videos.

There are several things that need to be done to work with structured FrameMaker. Much of what you do in the following exercises only needs to be done once. Rather than jumping into creating content in DITA, we'll start by building some supporting files. We'll set up FrameMaker for structure, build an EDD (based on the DITA rules), test the EDD, look at a sample document to map to the EDD, and create a conversion workflow. We'll evaluate the resulting files and eventually we'll be at the point a new author would be at. We'll have a template and a workflow and we can write. We'll take you behind the scenes to help you learn as much as possible about how to build and use DITA content with FrameMaker!

To get you started, we're going to work to build an EDD (although, technically, FrameMaker already has several to pick from) so that you can learn about structure from behind the scenes. To do so, it's important to configure FrameMaker to work with structured content. Then we'll build the EDD, which will contain rules with a similar structure as a basic DITA topic. While you can build an EDD for any structure, our topic-like DITA template allows our converted content to later be placed into the DITA templates included with FrameMaker. You can then see how much further a structured environment can take you.

Our goals are to give you a solid start when it comes to structure, AND to show you where you can go with structure.

Tips to get you started

Don't start by jumping right in and building an EDD and structured content. Instead review each of the tasks, scan them a second time, and then get started. You may also choose to watch our videos. If you really run into challenges, contact us and ask for sample files if it turns out you don't get you the results you expect.

Videos

The following hands-on materials are supported with "how-to" videos. Publishing Smarter has many videos to help you learn how to work with DITA. Look for them at **https://www.youtube.com/user/publishingsmarter/**

Visit the YouTube Playlists for specific videos related to the content that follows. Look for the *Migrate FrameMaker to Structured Content* area.

Sample content

All the content that follows can be created without the need to download specific sample content. Instead, all you need is a default installation of Adobe FrameMaker (2019 release).

Hands-on development of an EDD

- **Prerequisites**
- **Creating your first EDD**
- **Configuring your workspace**
- **Populate the EDD with the first element**
- **Build your own workspace**
- **Build the EDD content**
- **Test the EDD**
- **Assigning format rules for content using the EDD**
- **Formatting contextually**
- **Import and test the EDD formats**
- **Format a text string element**

Hands-on migration to structured content

- **Structured application files and XML round-tripping**
- **Implementing structured FrameMaker**
- **Migrating unstructured files to structure**
- **Tips and tricks about conversion to structure**

Prerequisites

Before building your first EDD ensure you are using the Structured Interface in FrameMaker. You should also be generally familiar with working in a structured interface using the Structure View and Element Catalog. If you've never done that, consider watching our sample videos.

Work with structured content as an author

While the EDD developer is expected to know how to work in structured content as an author, we know not everyone has had the chance to create content using the structured interface. We want you to learn how to build an EDD, which in itself is a structured document. As you build it you'll also start to learn to work within any structured environment. You'll have to use the Structure View, insert elements, add content, and make other changes that an author working with any structured content does.

Before you begin, take a few minutes and get familiar with the structured interface as a writer. There are a set of clear, short videos you should watch first. If you have already worked with structured content and are comfortable with how to author content in a structured workspace you can skip these.

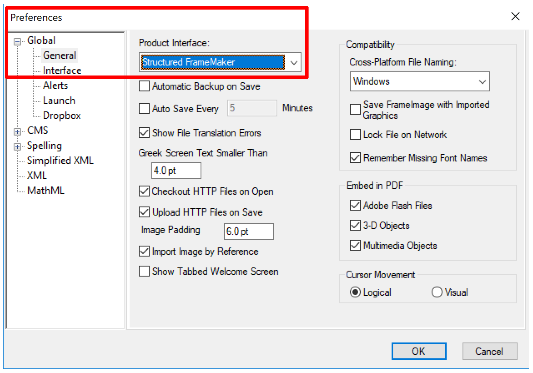
- Go to **https://www.youtube.com/watch?v=FwqC8gbH_Xc&list=PLdG-fUF32cbY3l8xCOMz4ZQeXjKb7tQTQE**
- Watch the videos and, if you prefer, create sample content as illustrated in the videos.

Configure the product interface

FrameMaker can be used in unstructured or structured formats. To create an EDD, or to work with structured content, you must switch to the Structured FrameMaker interface.

TIP: If you see menu options like **Element** or **Structure**, you're already there and can skip ahead to **Creating your first EDD**.

1. Select **Edit** > **Preferences**.
2. Under **Global** > **General** ensure you change the Product Interface to Structured FrameMaker.



Product Interface: Structured FrameMaker

□ Automatic Backup on Save

□ Auto Save Every 5 Minutes

☑ Show File Translation Errors

Greek Screen Text Smaller Than 4.0 pt

☑ Checkout HTTP Files on Open

☑ Upload HTTP Files on Save

Image Padding 6.0 pt

☑ Import Image by Reference

□ Show Tabbed Welcome Screen

Compatibility

Cross-Platform File Naming: Windows

□ Save FrameImage with Imported Graphics

□ Lock File on Network

☑ Remember Missing Font Names

Embed in PDF

☑ Adobe Flash Files

☑ 3-D Objects

☑ Multimedia Objects

Cursor Movement

● Logical ○ Visual

You can also turn off **Automatic Backup on Save** to reduce the number of files in any folder as you work with the tool.

3. Click **OK**, and if prompted, restart FrameMaker.

Creating your first EDD

It's simple to create the EDD, but developing it takes more work. Ensure you are in the structured interface and have completed the prerequisites. Illustrations are added to help clarify complex ideas. When in doubt, read the entire task, then review the images, and only then perform the actions. We've also create videos to help guide you through the process.

TIP: Pick a location for files and create a folder there. We'll create a folder named MyTutorials on the Desktop and reference it moving forward.

Generate an EDD and view boundaries and element structure

1. Select **Structure** > **EDD** > **New EDD**.

   The EDD is created. Default content is inserted by FrameMaker. It is a structured document, meaning as you learn to use the EDD you also learn to work with the structured interface!

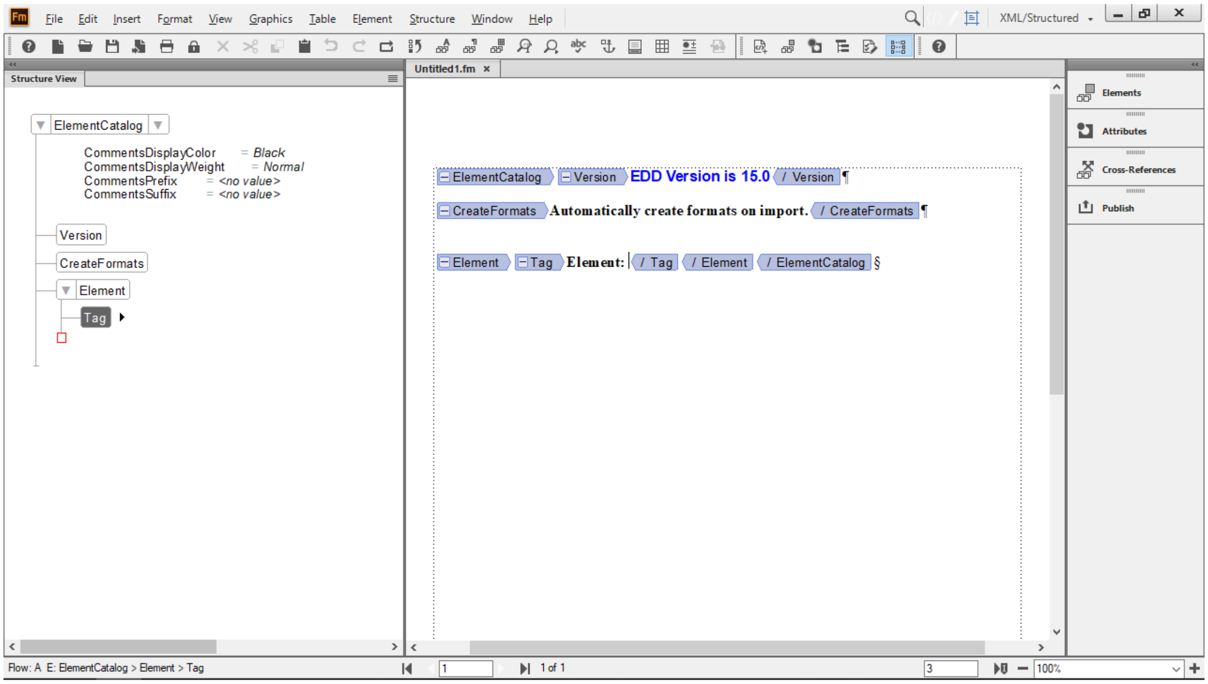
2. Select **File** > **Save As** and save the EDD as TopicEDD.fm in one folder that you will use for all your hands-on work.

   For example, save the EDD to the folder named "MyTutorials" on your desktop.

3. Select **View** > **Element Boundaries (As Tags)**.

4. If the interface does not appear as seen in Figure 27 don't panic. Continue to **Configuring your workspace**.

Figure 27: XML/Structured Interface



ElementCatalog

CommentsDisplayColor = *Black*
CommentsDisplayWeight = *Normal*
CommentsPrefix = *<no value>*
CommentsSuffix = *<no value>*

Version
CreateFormats
Element
Tag

ElementCatalog Version EDD Version is 15.0 / Version ¶

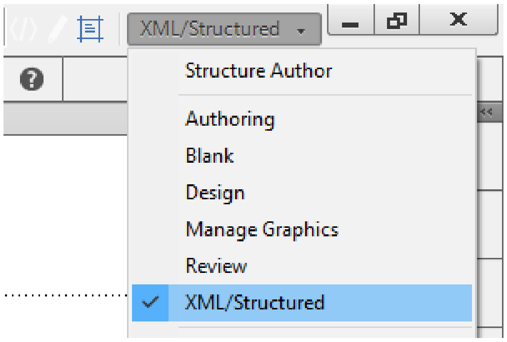
CreateFormats Automatically create formats on import. / CreateFormats ¶

Element Tag Element: / Tag / Element / ElementCatalog §

Flow: A E: ElementCatalog > Element > Tag   1 of 1   100%

Configuring your workspace

To get the most out of working with an EDD (or any structured content) the workspace you use can be configured to help guide you. This includes using the Structure View, the Element Catalog, and working with Attributes. T



Depending on your screen resolution it may be easier or more involved to have all the pods we suggest opened. In this section we'll explore both suggested minimum and suggested ideal ways to have things set up on screen. At a minimum, the icons for common EDD development tools should be quickly available. Adobe provides a default workspace for this.

Assign a default workspace

1. In the top right of FrameMaker, click the **Workspace** dropdown.

2. Choose *XML/Structured*.

Once done, icons for the Structure View, Element Catalog, and Attributes Dialog display. If not, you can select the same dropdown and Reset workspace.

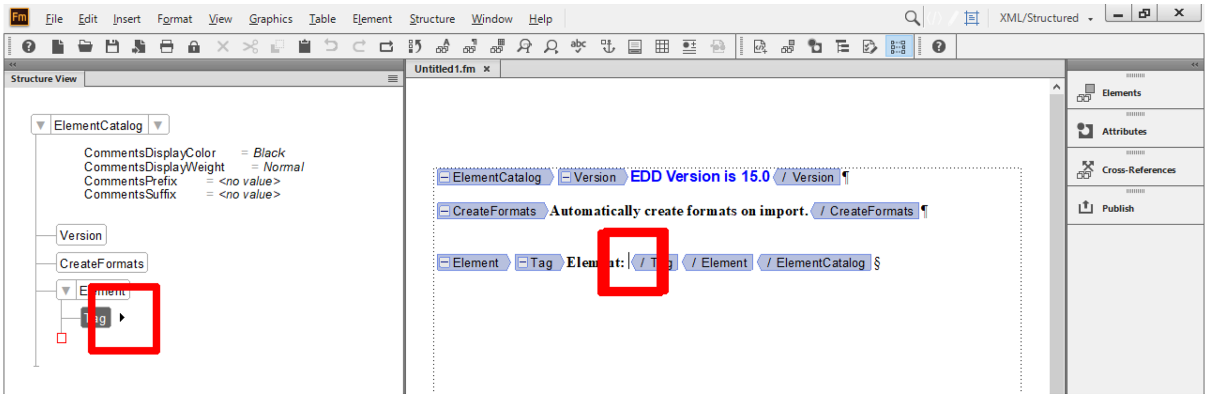
Later, we'll build a new workspace from scratch.

Populate the EDD with the first element

The basic structure for our topic needs to be developed. If you were working with an existing EDD this would already be done, but it's a good way to get to know a bit about the organization of our structure and to work with the Structured Interface.

1. Ensure your insertion point is positioned in the element named **Tag**.

   Do not click the name/bubble of the element Tag. You can either click to the right of the bubble named Tag in the Structure View, or between the Tag markup in the document view (next to the word **Element:**).


File Edit Insert Format View Graphics Table Element Structure Window Help
XML/Structured
Structure View
Untitled1.fm
ElementCatalog
CommentsDisplayColor = Black
CommentsDisplayWeight = Normal
CommentsPrefix = <no value>
CommentsSuffix = <no value>
Version
CreateFormats
Element
Tag
ElementCatalog Version EDD Version is 15.0 / Version ¶
CreateFormats Automatically create formats on import. / CreateFormats ¶
Element Tag Element: / Tag / Element / ElementCatalog §
Elements
Attributes
Cross-References
Publish

2. Type the word topic.

As you type, the content appears in both the Structure View and the document window.

Build your own workspace

Configuring the placement of components like the Element Catalog and Structure View allows you to create and save a custom workspace.

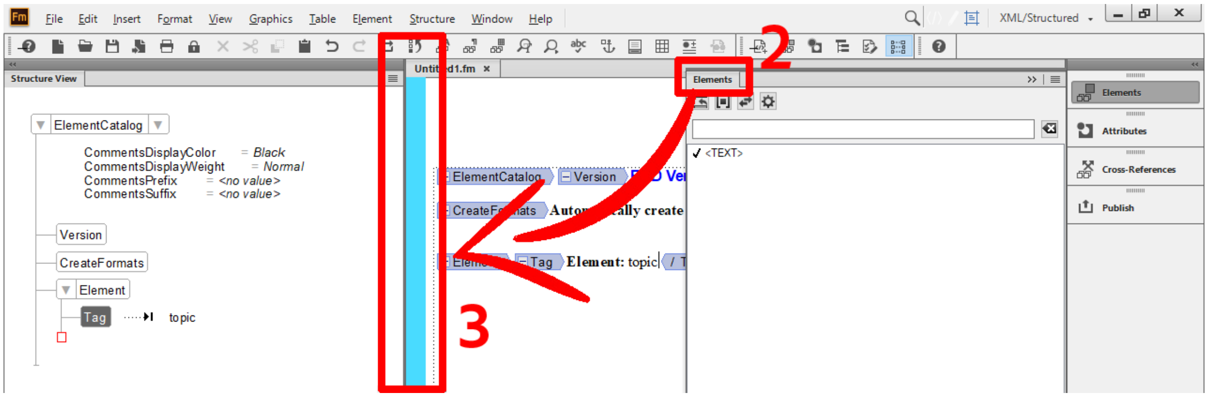
Before you begin, you may need to click the Workspace dropdown and select Blank. You may also need to reset it if components are visible.

1. Select Structure > Structure View and, once the Structure View is displayed, drag and drop it to the left of the application window.

   You may need to move it slowly to the side and wait for a highlight to appear (and the tab to fade) before dropping it.

2. Select View > Pods > Element Catalog, then move the pod between the Structure View and the document.

3. When the highlight displays, drop the pod.


File Edit Insert Format View Graphics Table Element Structure Window Help
XML/Structured
Structure View
Untitled1.fm
Elements
ElementCatalog
CommentsDisplayColor = Black
CommentsDisplayWeight = Normal
CommentsPrefix = <no value>
CommentsSuffix = <no value>
Version
CreateFormats
Element
Tag
topic
ElementCatalog Version EDD Ver
CreateFormats Automatically create
Element Tag Element: topic / T
✓ <TEXT>
2
3
Elements
Attributes
Cross-References
Publish

4. Reorganize the workspace as required based on your resolution and needs.

   This may include resizing or changing the zoom of various parts of the FrameMaker interface. Configure it so that it appears similar to what is seen here.

5. When done, click the **Workspace** dropdown at the top right of the screen and choose **Save workspace**.
6. Name the workspace **Structured Developer**.
7. Click **OK**.

Build the EDD content

The remainder of the EDD drives the entire structure and will be tested later.

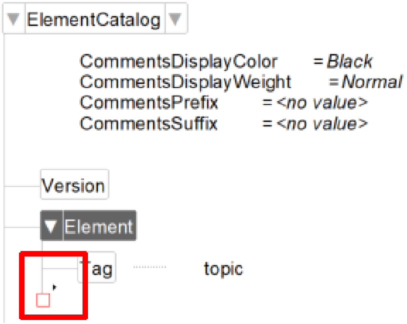
Develop the highest level <topic> element

While the tag named topic has been started, there are still parts of it missing. FrameMaker guides you through the requirements for structured content.

1. Select **View**, then deselect **Element Boundaries (as Tags)**.

   This simplifies the interface.

2. Click to the right and just above the red box in the Structure View.



CommentsDisplayColor = *Black*
CommentsDisplayWeight = *Normal*
CommentsPrefix = *<no value>*
CommentsSuffix = *<no value>*

   The Elements catalog updates to show valid elements. If you do NOT see an element named **Container** you may have to choose Options (the Gear icon) and select Valid Elements for Working Start to Finish.

3. In the Elements catalog double-click **Container**.

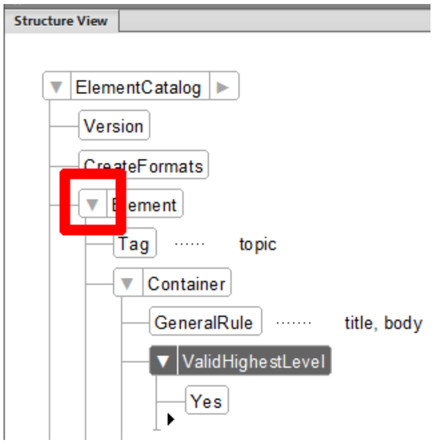
   Both the **Container** and the **GeneralRule** are inserted.

4. Type **title, body**
5. In the Structure View, click immediately below the **GeneralRule**.
6. In the Elements catalog double-click **ValidHighestLevel**.

   The topic element has been developed.

7. Collapse the Element by clicking the **triangle** pointing down and toggling it.

   You can collapse content in the Structure View by clicking triangle as needed. For example, clicking it to the left of **Element** collapses the element to preserve space in the Structure View. This is helpful when many elements are visible.



Add a <title> element

Based on our earlier content analysis the structure needs further definition. We'll add a title element using simpler instructions (hopefully it's getting a bit easier).

1. Click immediately below the collapsed **Element** tag.
2. Insert another **Element** (remember to double-click it in the Elements catalog).

   The insertion point is already immediately next to the element in the Structure View, so you can begin to type the name of the element immediately).

3. Type title

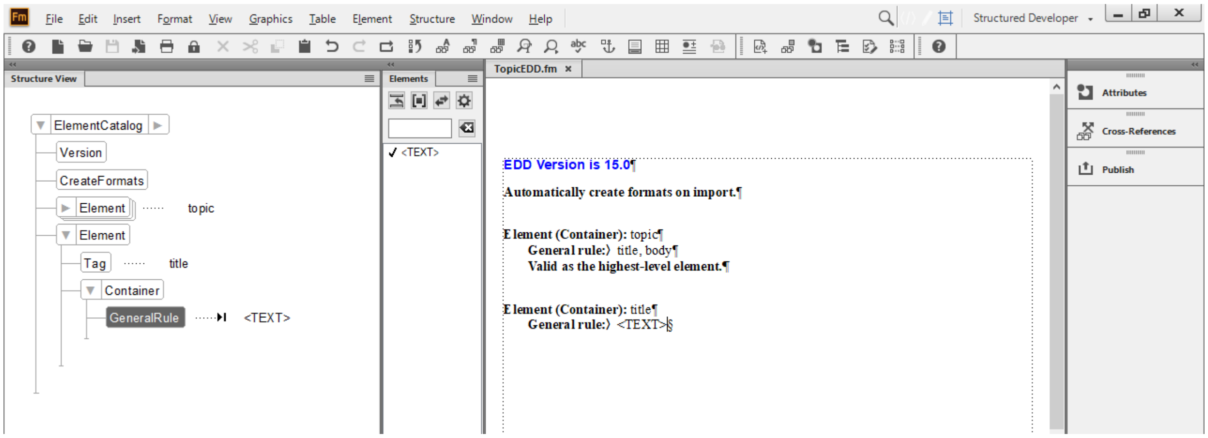
4. Click to the right and just above the red box in the Structure View.

5. Insert a Container (remember to double-click it in the Elements catalog).

   Both the Container and the GeneralRule are inserted.

6. Type <TEXT>

   This rule allows users to type in this element when working as authors.



EDD Version is 15.0¶

Automatically create formats on import.¶

Element (Container): topic¶
General rule:) title, body¶
Valid as the highest-level element.¶

Element (Container): title¶
General rule:) <TEXT>§

7. Collapse the Element by clicking the Downward Triangle.

Add a <body> element

Let's speed up a bit more and add a body element below the most recent content.

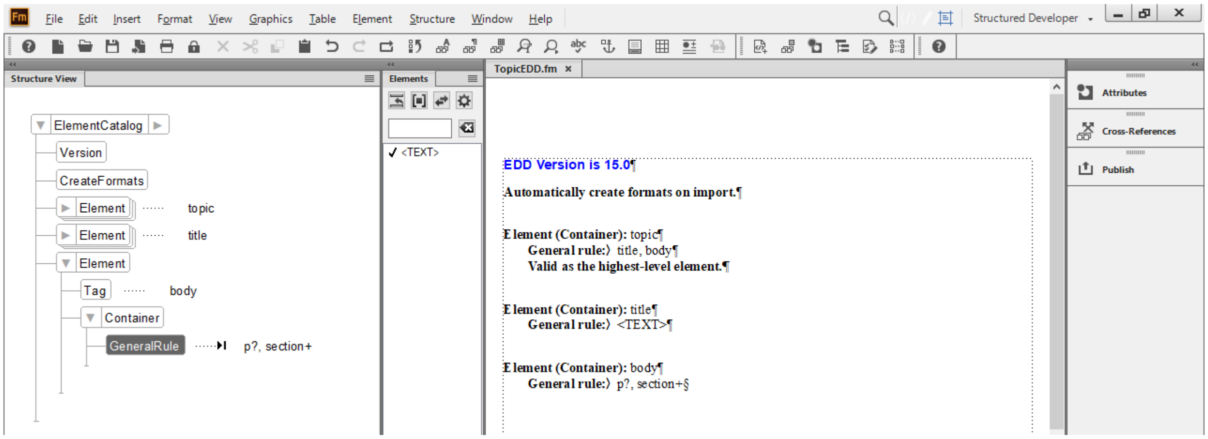
1. Click immediately below the collapsed Element tag.

2. Insert another Element and in the Tag type body

3. Click to the right and just above the red box in the Structure View and insert a Container.

4. Type p?, section+

   This rule allows authors to insert an optional <p> element, followed by one or more <section> elements.



EDD Version is 15.0¶

Automatically create formats on import.¶

Element (Container): topic¶
General rule:) title, body¶
Valid as the highest-level element.¶

Element (Container): title¶
General rule:) <TEXT>¶

Element (Container): body¶
General rule:) p?, section+§

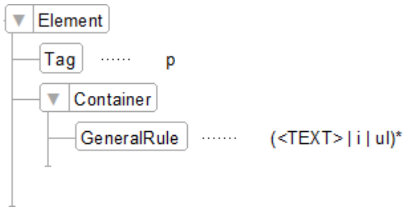
5. Collapse the Element.

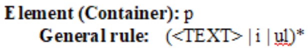
Add a <p> element

We'll speed up the creation of this element even more!

1. Click immediately below the collapsed **Element** tag and insert an element, naming it p
2. Insert a Container.
3. Type (<TEXT> | i | ul)*

   Include the parenthesis. This rule allows authors to insert text, italic content, or an unordered list.
4. Collapse the **Element**.



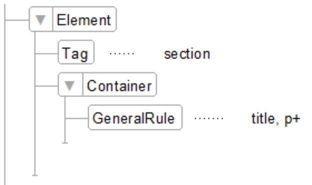
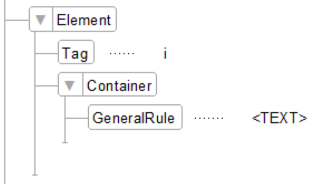
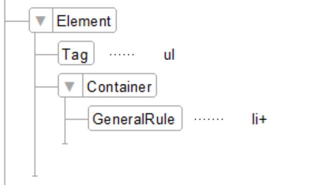
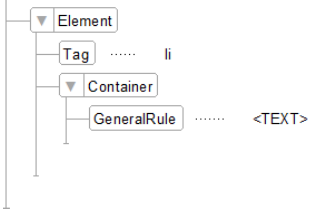


Element (Container): p
General rule: (<TEXT> | i | ul)*

Complete the EDD structure

Finish the following steps and apply what you've learned to this point to complete the structure.

1. Complete the structural rules by adding the elements seen here.

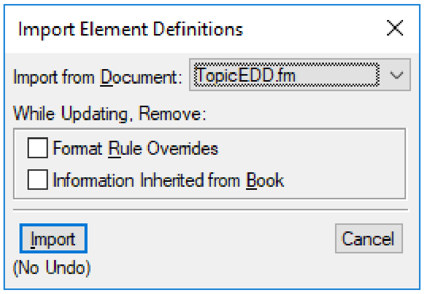
Element	General Rule	Notes	Illustration of result
section	title, p+	The <section> contains a required title (but only one, as well as one or more <p> elements.	 Element Tag section Container GeneralRule title, p+
i	<TEXT>	The <i> element is for content that will be formatted (later) as italic and it contains text only.	
ul	li+	The unordered list contains one or more list items.	
li	<TEXT>	Each list item contains text.	

The EDD now contains the structure you want for the topic, but with no formatting. Before adding formatting information, it's a good idea to test the structure.

Test the EDD

To test the EDD, import it to a document. Verify you can create the expected structure. It is important to remember that **in this part of the exercise no formatting is applied!** The goal is only to validate the structure works.

1. Select File > New > Document and click Portrait.
2. Select File > Import > Element Definitions.
3. Ensure that TopicEDD.fm is selected under Import from Document.



Import Element Definitions

Import from Document: TopicEDD.fm

While Updating, Remove:

□ Format Rule Overrides

□ Information Inherited from Book

4. Click Import.

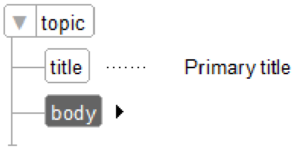
   Element definitions import. A successful log file can be closed. If there is an error review the previous steps. Errors may be due to an invalid source EDD. If so, compare your EDD with that from Creating your first EDD. Troubleshooting links from the error report to the EDD are created. Structure definitions in the EDD are imported to the blank document.

5. To verify that the definitions were imported, click once in the main text flow and look at the Elements catalog.

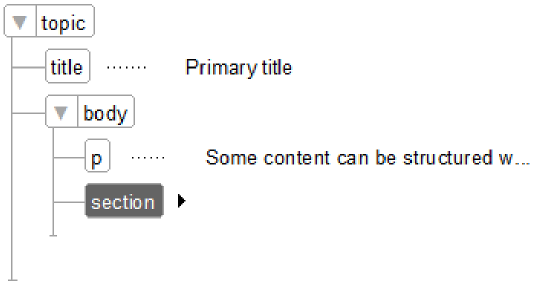
   You should see the topic element.

6. Insert a topic element.
7. Insert a title element.
8. In the title type the text Primary Title.
9. Insert a body element.

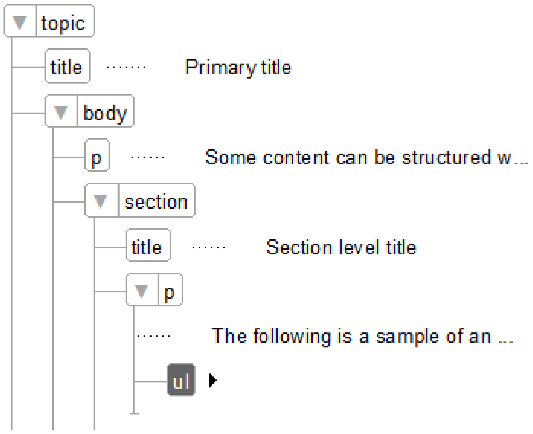
   The structure should appear as seen here.



10. Within the body add a child p element.
11. In the p type Some content can be structured with character level formatting.
12. Equal to the p element add a section element.



13. In the section element add a title.
14. In the title type the text Section level title.
15. Equal to the title element add a p element.
16. In the p element type The following is a sample of an unordered list.
17. At the end of the text in the p element insert a ul element.

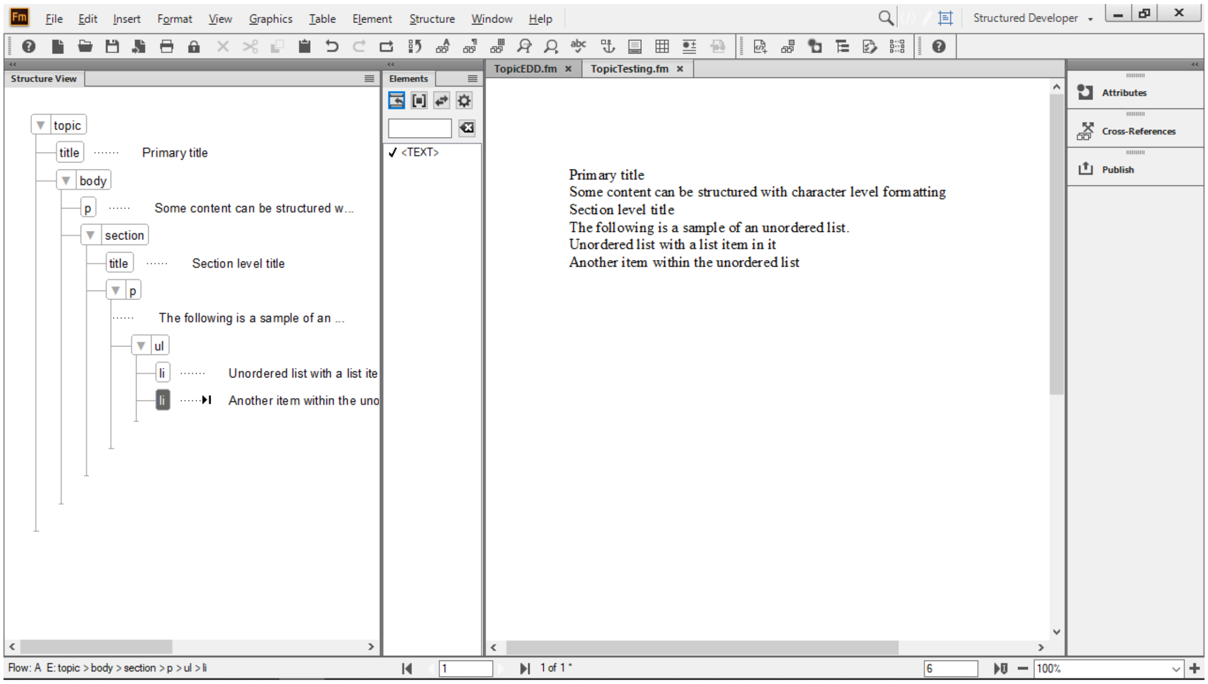

topic
title
Primary title
body
p
Some content can be structured w...
section
title
Section level title
p
The following is a sample of an ...
ul

18. In the ul element insert two li elements.

19. In the first li type Unordered list with a list item in it.

20. In the second li type Another item within the unordered list.

21. Save your file as *TopicTesting.fm* in the same *folder* as your TopicEDD.fm.



Primary title
Some content can be structured with character level formatting
Section level title
The following is a sample of an unordered list.
Unordered list with a list item in it
Another item within the unordered list

Flow: A E:topic > body > section > p > ul > li

Assigning format rules for content using the EDD

The EDD you built provides structure for a simple topic. However, when you insert content, no formatting is applied. Working with your EDD and template you can automatically apply the correct format for the various titles.

By default, text uses the Body paragraph tag. We'll update the list items to appear as bullets, the titles to appear as either a Title or a Heading1 based on the context in which they are used, and assign character-level formatting.

Create a format for list items

In our EDD we have a rule that the unordered list contains one or more li elements. The li elements contain text, but have no format rules. We'll update this and import the rules to our test file.

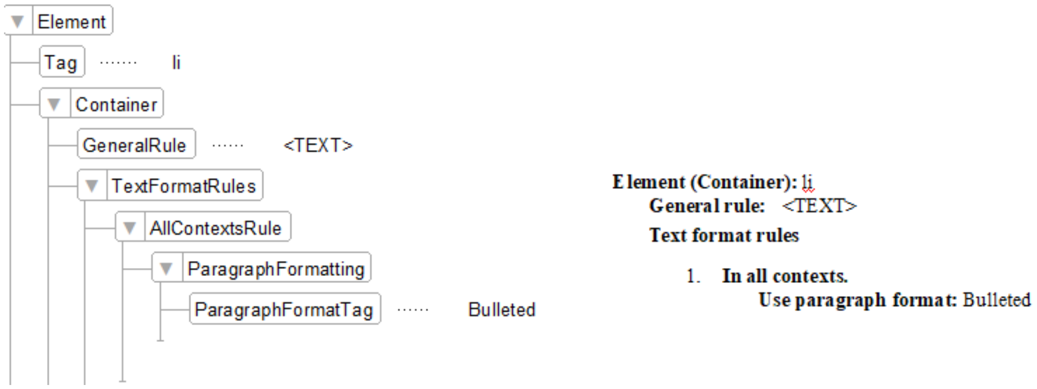
1. Switch to your EDD.

2. Within the element tagged as li, equal to the GeneralRule, add a TextFormatRules element with nested children named AllContextsRule, ParagraphFormatting, and ParagraphFormatTag.

3. Type Bulleted as the text in ParagraphFormatTag.

   The finished structure is seen in Figure 28.

Figure 28: Final <li> format for a list item

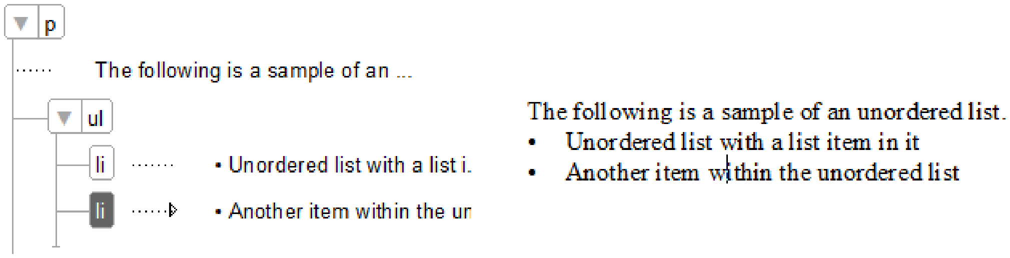


Element (Container): li
General rule: <TEXT>
Text format rules

1. In all contexts.
Use paragraph format: Bulleted

Import the new format

1. Switch to the *TopicTesting.fm* file.
2. Select **File** > **Import** > **Element Definitions**.
3. Ensure that *TopicEDD.fm* is selected as the source to import from.
4. Click **Import**.
The element definitions are imported successfully. The format of the two **li** elements changes and bullets are visible in the document view. The structure remains the same.


p
The following is a sample of an ...
ul
li
• Unordered list with a list i.
li
• Another item within the ur

The following is a sample of an unordered list.
- Unordered list with a list item in it
- Another item within the unordered list

Formatting contextually

Rather than formatting content the same way in all cases, you can define contextual rules. For example, you may base format on the parent element and assign 2 formats to the same element, but depend on the parent to change the context.

This allows you to build multiple unique and contextually driven formats. For example, consider where a list item may appear. If in a parent of <ul> or <ol> you may want to format the list item with bullets or numeric values. Alternatively, A title could appear in many places and therefore has contextual formatting. For example, in a <topic> element it may appear as a Title, and within a <section> element as a Heading1.

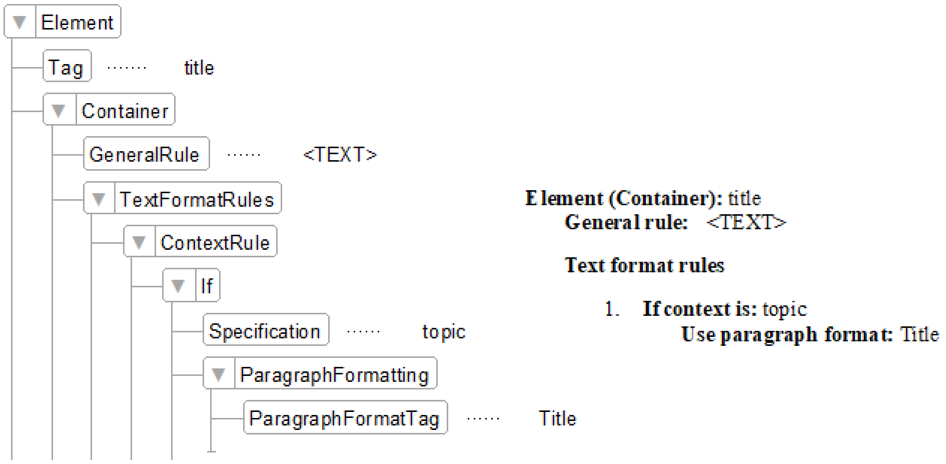
Contextually format a topic <title> element

The **title** requires a more complex rule. It can display in either the **topic** element, or in the **section** element. Therefore, a context rule is required. Review **Figure 29** before attempting the steps to gain a better understanding of the finished structure.

1. Switch to your EDD.
2. Build the structure in **Figure 29**.

   If required, the steps in **Assign a topic title format (detailed steps)** detail how to create this structure and appear below **Figure 29**. Otherwise, proceed to **Format a section <title> element**.

Figure 29: Final <title> format for a topic title



```
Element (Container): title
    General rule:   <TEXT>

    Text format rules

        1.  If context is: topic
                Use paragraph format: Title
```

Assign a topic title format (detailed steps)

Only perform these steps if you have not already completed the work illustrated in Figure 29. If this was already done, skip to **Format a section <title> element**.

1. In the element tagged as **title**, after the **GeneralRule**, add a **TextFormatRules** element with nested children named **ContextRule**, **If** and **Specification**.

2. In **Specification** type topic.

3. Below the Specification insert a **ParagraphFormatting** and **ParagraphFormatTag**.

4. Type Title as the text in **ParagraphFormatTag**.

   This is the first of two rules. It states that if a **title** is within the **topic** the paragraph format to apply is Title (note the uppercase initial character as it matches the format in the Paragraph Catalog in our TopicTesting file).

Format a section <title> element

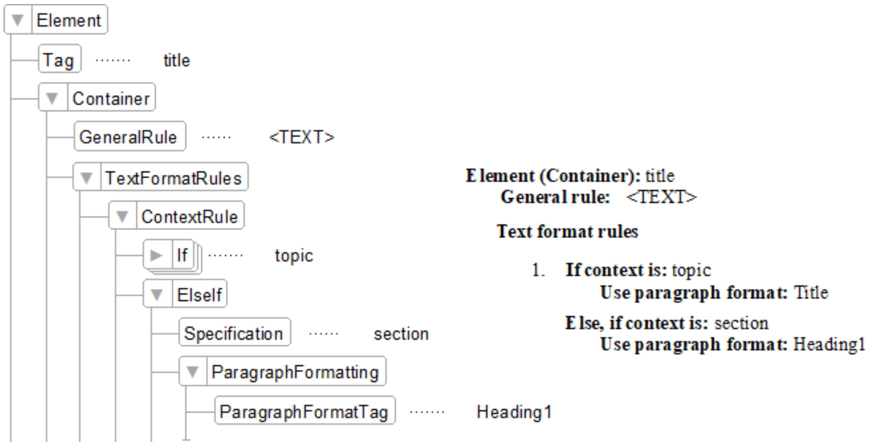
1. Click the collapsing triangle to the left of the **if** element to collapse it.

2. Immediately below the collapsed **if** element add an **ElseIf** and **Specification**.

3. Complete the structure to appear as seen in **Figure 30**.

   If required, **Assign a section title format (detailed steps)** details this structure and appears below **Figure 30**. Otherwise, proceed to **Import and test the EDD formats**.

Figure 30: Final <title> format for a section



```
Element (Container): title
    General rule:   <TEXT>

    Text format rules

        1.  If context is: topic
                Use paragraph format: Title
            Else, if context is: section
                Use paragraph format: Heading1
```

Assign a section title format (detailed steps)

Only perform these steps if you have not already completed the work illustrated in Figure 30. If this was already done, skip to Import and test the EDD formats.

1. In the element tagged as **title**, add a **TextFormatRules** element with nested children named **ContextRule**, **ElseIf** and **Specification**.
2. In **Specification** type section.
3. Below the **Specification** insert a **ParagraphFormatting** and **ParagraphFormatTag**.
4. Type Title as the text in **ParagraphFormatTag**.

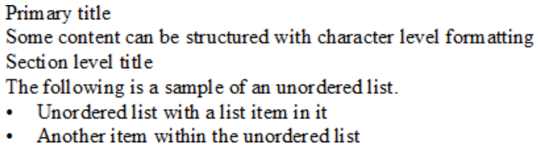
   This is the second of two rules. It states that a **title** within the **section** is formatted as a Heading1 paragraph (note the uppercase initial character, and the lack of a space before the number as it matches the format in the Paragraph Catalog in our TopicTesting file).
5. Click the collapsing triangle to the left of the **ElseIf** element to collapse it.

Import and test the EDD formats

Once format rules have been defined they can be imported and tested in a structured file.

1. Switch to the *TopicTesting.fm* file.
2. Review the current format of the text Primary title and of the text Section level title.

   The text appears in the Structure View and in the document. Both titles are formatted as Body paragraphs by default. Once format rules are imported the appearance of both title elements will change based on context in the structure.

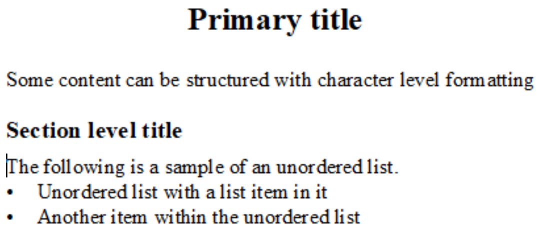


Primary title
Some content can be structured with character level formatting
Section level title
The following is a sample of an unordered list.

- Unordered list with a list item in it
- Another item within the unordered list

3. Select **File** > **Import** > **Element Definitions**.
4. Ensure that *TopicEDD.fm* is selected as the source to import from.
5. Click **Import**.

   Element definitions are imported successfully. The format of the two **title** elements changes and formats are visible in the document view. The structure remains the same.



Primary title

Some content can be structured with character level formatting

Section level title

The following is a sample of an unordered list.

- Unordered list with a list item in it
- Another item within the unordered list

Format a text string element

Content can be structured and formatted at a character level as well. To illustrate this we'll insert a string of text and structure it. Then we'll update the EDD and import it one more time.

1. In your document, locate the text Some content can be structured with character level formatting.
2. Double click the word character.
3. In the Element Catalog double click the element **i**.

   This wraps the content in the element used to represent italics. As no character format is yet assigned the element breaks to the next line. Structurally the content is valid.
4. Switch to your EDD.
5. In the element tagged as **i** add a **TextFormatRules** element with nested children named

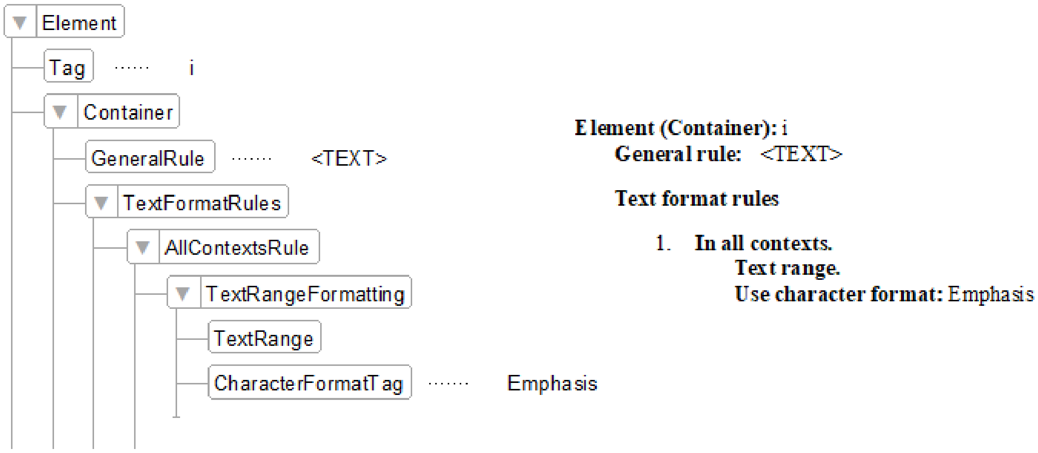
AllContextsRule, TextRangeFormatting, and TextRange.

6. Immediately below the TextRange insert the CharacterFormatTag element.

7. Type Emphasis as the text in CharacterFormatTag.

   The finished structure also updates the document window for the i element.

Figure 31: Completed character format



Element (Container): i
General rule: <TEXT>

Text format rules

1. In all contexts.
   Text range.
   Use character format: Emphasis

8. Switch to the TopicTesting.fm file.

9. Select File > Import > Element Definitions and, selecting the TopicEDD.fm, click Import. The element definitions are imported successfully. Since we defined the element as a TextRange the format of the i element changes and it appears inline and formatted. The structure remains the same, but the text is now treated as a string of text.

10. Save both open files and close them.

Further formatting

In a more complex EDD, the formatting would have to be further defined. There are many rules on how to format a document. For more information, refer to the Adobe documentation on developing formatting in an EDD. More complex content (for example, tasks with steps, multiple levels of bullets, tables and images, cross references, and other FrameMaker content) usually means you also need more complex EDDs and more format rules.

Further testing

If you wish, continue to add elements or text to your structure and ensure the formats appear as expected. You can reorganize elements within the context of what is valid (or break your structure), dynamically reformat content (try to switch out the titles of a section and the topic by dragging/dropping them in the Structure View). More complex structures also require more complex testing for both structure and format.

Structured application files and XML round-tripping

Since we are focused on migrating content to structure we're creating content that we can add to the default DITA environment in FrameMaker. To that end, we've ensured that you know more about the EDD and how to assign formatting. There are also application files that are used by FrameMaker to reference entire workflows for structure. A complete set of DITA-based applications already exist.

An application is used to convert pure XML content into a FrameMaker friendly document. This application can then be used to round-trip XML content to and from FrameMaker. If you build an application file there is testing to be done to ensure it works as expected. FrameMaker includes a complete set of DITA-based round-tripping applications that have been extensively tested by Adobe. Therefore, we do not need to test the XML round-trip.

Implementing structured FrameMaker

Any implementation is going to take time, have unique challenges, and require a broad mix of skills. We'll help prepare you for the implementation, but there is going to be a lot of planning and testing before things work as you expect them to.

Analyze your content

Before building your own structured environment, and documents like an EDD, it's helpful to do a detailed content analysis. Content analysis is a process in which existing documents are reviewed to understand their design and content and what implicit structure they contain. This helps to determine what is required, optional, and how often it occurs.

Begin your analysis by making a list of the documents your organization produces. This list might include user guides, reference guides, whitepapers, tutorials, training manuals, and online help. Make a list of the major components for each document type.

Table 9 shows, at a high level, what you may find within your own content (a cover page, legal notes, a table of contents, chapters, and an index). There may also be optional material (a foreword, list of figures, list of tables, appendices, and a list of authors). Some content may occur only once (cover, table of contents, index) and other content may occur one or more times (chapters or appendices).

Table 9: Initial sample analysis of content frequency

COMPONENT	REQUIRED OR OPTIONAL?	OCCURRENCE
Cover	Required	1
Legal Notes	Required	1
Table of contents	Required	1
List of Figures	Optional	0 or 1
Chapter	Required	1 or more
Appendix	Optional	0 or more
Glossary	Optional	0 or 1
Index	Required	1

For each of the major components, such as modules (which may be part of your training manuals) or chapters (such as those in traditional books), work your way down the document hierarchy. You'll start to list smaller and smaller chunks until you reach the bottom level of the hierarchy. Eventually you end up with index entries, character level objects (often things like a menu option [File, Insert, View, etc], specific keywords [Product Name, Branded Phrase, etc], window titles [Insert Table Dialog, Configure Settings Dialog, etc], or others), and even hard formats (such as Emphasis, or Bold).

Documents are unlikely to be perfect in consistency, so a decision must be made around the structure and how "loose" or "strict" it will be. Each has potential benefits and drawbacks:

- Loose allows for wide variations (think about how many ways people construct content for websites) and can be complex to maintain, but allows writers broad freedoms.

- Strict is very limited (think about how formal a database of parts could be, or the simplicity of a glossary with terms and definitions) and is simpler to maintain, but restricts possible variations.

Neither is "right" nor "wrong" and, instead, you will likely land somewhere between the two. Of course, this same issue exists with templates created in an unstructured workflow. Too many specific tags for characters, tables, and paragraphs and writers spend all their time trying to remember which to apply; not enough tags and writers create content in the broadest way possible.

Decide on a structured approach

Post-analysis you need to decide if an existing structured standard will work, or if you need to build your own. Standards help get you started, provide a baseline that may already be supported in your tools, and could also be required in specific areas (medical, aerospace, or military for example). Building your own has more moving parts, but gives you exactly what you need. We'll get you working hands-on with DITA shortly, but let us first compare why you may want to use an existing standard or build your own.

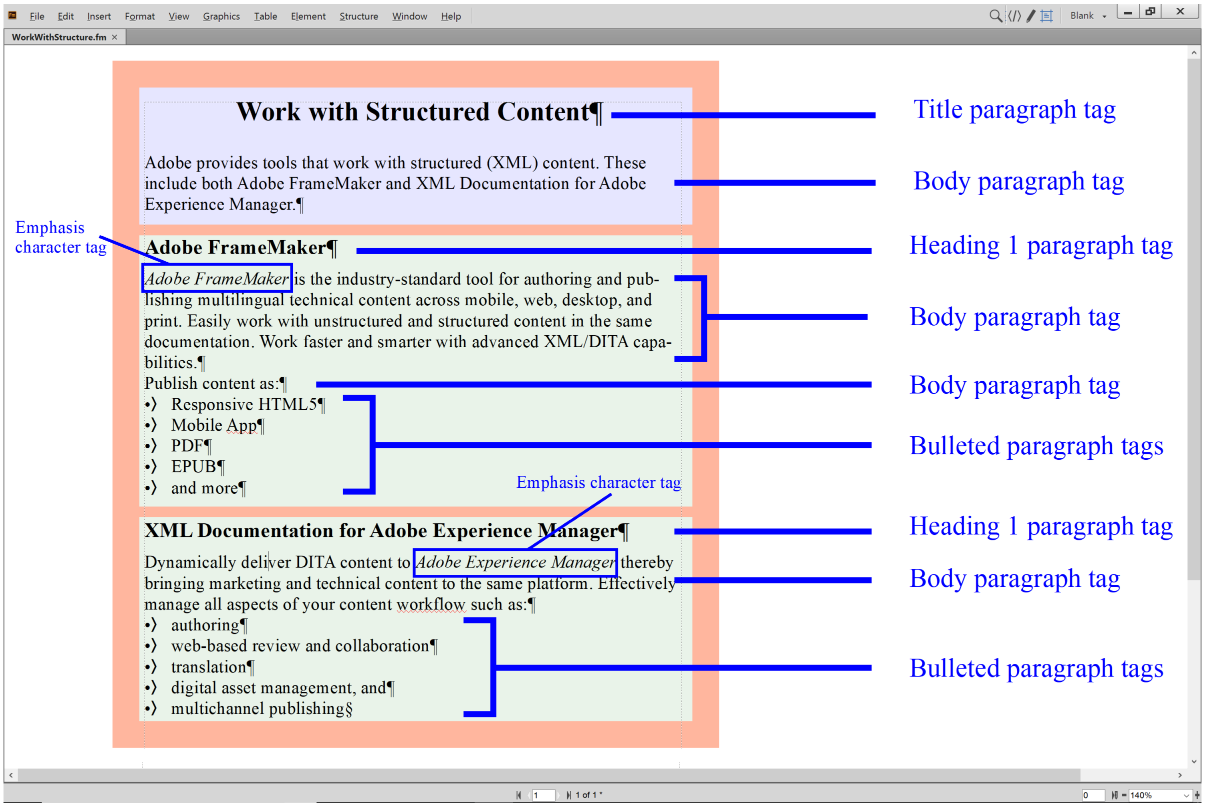
Table 10: Factors that influence using standards or customizing

Use a standard (DITA, S1000D, DocBook, etc)	Build your own (Custom rules)
Regulatory: You must create content that follows the standard (aerospace, military)	You want a structure that matches your specific content analysis
Content is already close to a standard, with minimal overall effort you could use one that exists	Content analysis resulted in content that does not mach any standard, and there is little or no flexibility to change
Less effort in standards development; instead, you put time and effort into conversion, rework, and layout/design	Structure must match and putting time into building a formal set of rules, plus the work in conversion and layout/design, is acceptable
Technical skills to do the work are not available internally	Technically experts are available and can build your custom structure

Sample analysis of a document (illustrated)

There isn't enough time to create dozens of files to review. Instead, let's look at one simple document. It's already marked up, but it's easy to create the following if needed in FrameMaker. (We'll actually walk you through this file later on.) The collected content will be one document type we'll call a **topic**.

Figure 32: Sample topic with tagging



Work with Structured Content¶

Adobe provides tools that work with structured (XML) content. These include both Adobe FrameMaker and XML Documentation for Adobe Experience Manager.¶

Adobe FrameMaker¶

Adobe FrameMaker is the industry-standard tool for authoring and publishing multilingual technical content across mobile, web, desktop, and print. Easily work with unstructured and structured content in the same documentation. Work faster and smarter with advanced XML/DITA capabilities.¶

Publish content as:¶

- Responsive HTML5¶
- Mobile App¶
- PDF¶
- EPUB¶
- and more¶

XML Documentation for Adobe Experience Manager¶

Dynamically deliver DITA content to *Adobe Experience Manager* thereby bringing marketing and technical content to the same platform. Effectively manage all aspects of your content workflow such as:¶

- authoring¶
- web-based review and collaboration¶
- translation¶
- digital asset management, and¶
- multichannel publishing§

Based on Figure 32, we can perform some basic content analysis. The entire document is a topic with a primary title and a paragraph. There are two sections. Each section has a Heading1 followed by more paragraphs. Some paragraphs are followed by bulleted lists. There is also some character formatting applied.

Create representative unstructured content

For the purpose of this exercise, create the content as seen in Figure 32. This is the same content we've already analyzed.

1. Select **File** > **New** > **Document**, then click **Portrait**.
2. Type in the following content and leave it formatted as Body paragraphs:

   Work with Structured Content¶
   Adobe provides tools that work with structured (XML) content. These include both Adobe FrameMaker and Adobe Experience Manager Guides.¶
   Adobe FrameMaker¶
   Adobe FrameMaker is the industry-standard tool for authoring and publishing multilingual technical content across mobile, web, desktop, and print. Easily work with unstructured and structured content in the same documentation. Work faster and smarter with advanced XML/DITA capabilities.¶
   Publish content as:¶
   Responsive HTML5¶
   Mobile App¶
   PDF¶
   EPUB¶
   and more¶
   Adobe Experience Manager Guides¶
   Dynamically deliver DITA content to Adobe Experience Manager thereby bringing marketing and technical content to the same platform. Effectively manage all aspects of your content workflow such as:¶
   authoring¶
   web-based review and collaboration¶
   translation¶
   digital asset management, and¶ multichannel publishing¶

3. Format content as Title, Body, Heading1, and Bulleted paragraphs based on Figure 32.

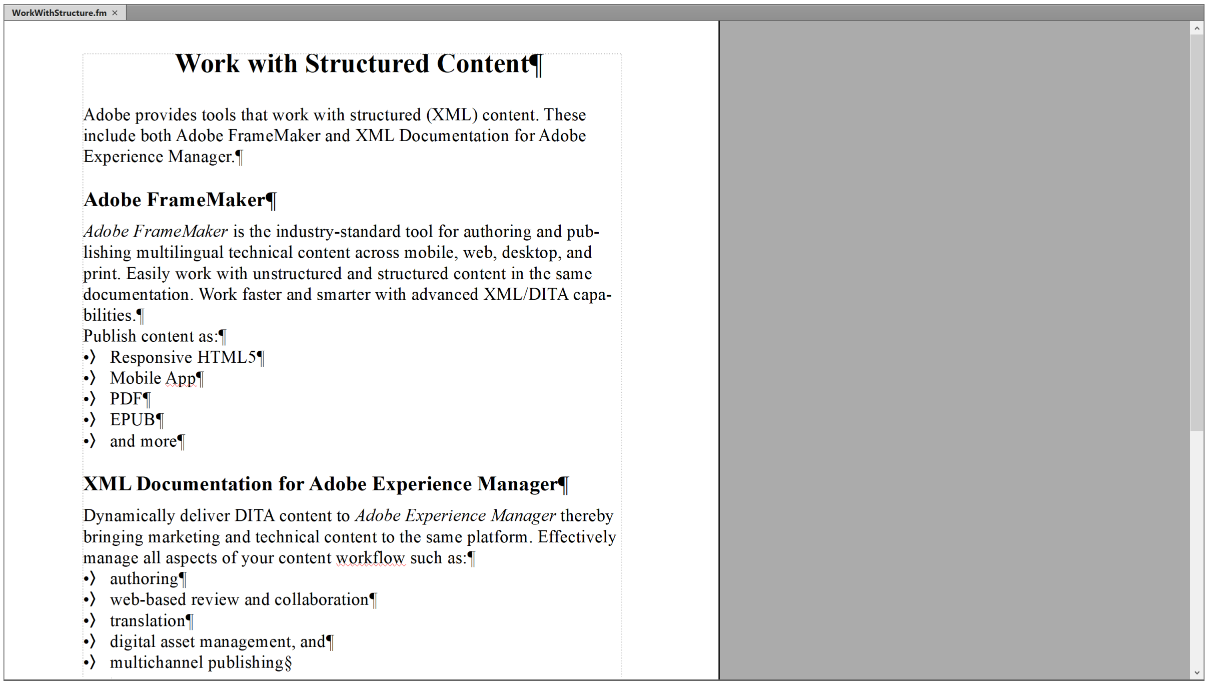
   For example, the Title is applied to the text *Work with Structured Content*, and so on.

4. Within the Heading1 titled *Adobe FrameMaker* select the body text content Adobe FrameMaker and apply the Emphasis character format.

5. Within the Heading1 titled *Adobe Experience Manager Guides* select the body text content Adobe Experience Manager and apply the Emphasis character format.

6. Select **File** > **Save As** and save the file as WorkWithStructure.fm in a location you can easily find later.
   For example, use the Desktop folder named MyTutorials (created earlier). The document should appear as seen in Figure 33.

Figure 33: Unstructured sample content



Work with Structured Content¶

Adobe provides tools that work with structured (XML) content. These include both Adobe FrameMaker and XML Documentation for Adobe Experience Manager.¶

Adobe FrameMaker¶

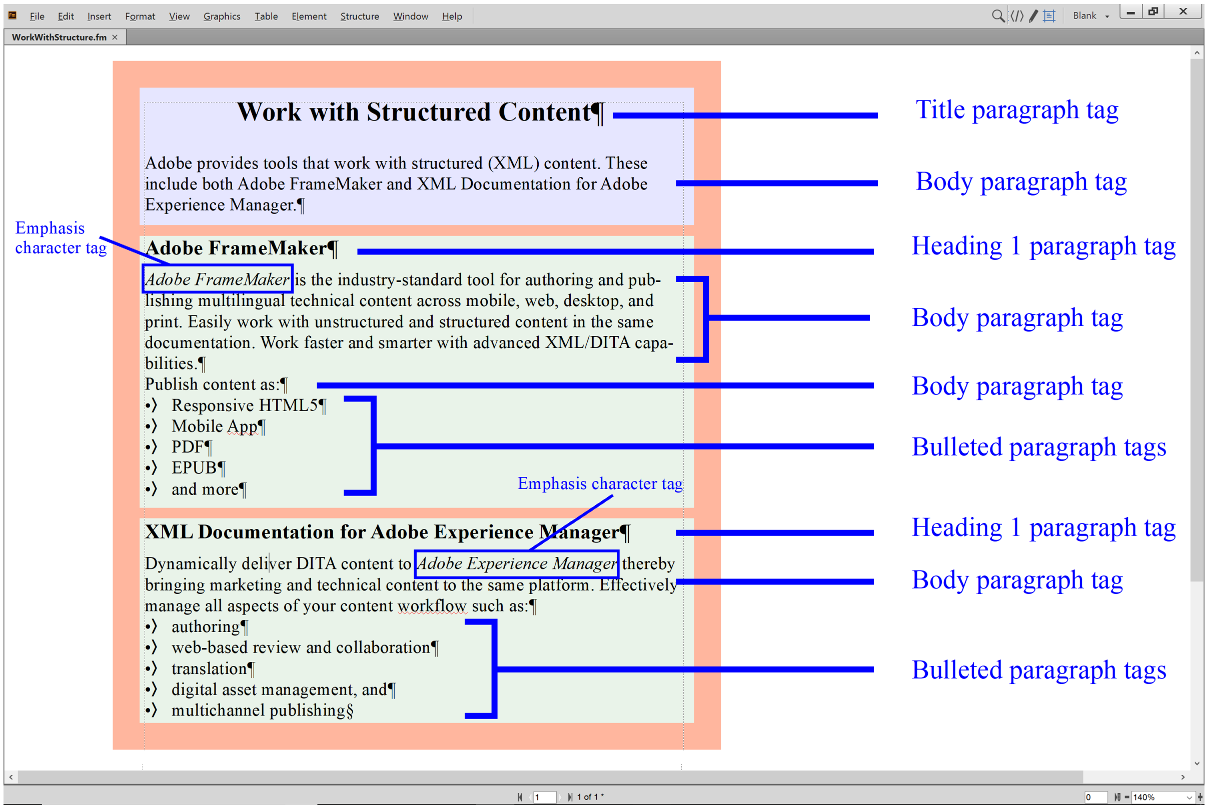
Adobe FrameMaker is the industry-standard tool for authoring and publishing multilingual technical content across mobile, web, desktop, and print. Easily work with unstructured and structured content in the same documentation. Work faster and smarter with advanced XML/DITA capabilities.¶
Publish content as:¶
•) Responsive HTML5¶
•) Mobile App¶
•) PDF¶
•) EPUB¶
•) and more¶

XML Documentation for Adobe Experience Manager¶

Dynamically deliver DITA content to *Adobe Experience Manager* thereby bringing marketing and technical content to the same platform. Effectively manage all aspects of your content workflow such as:¶
•) authoring¶
•) web-based review and collaboration¶
•) translation¶
•) digital asset management, and¶
•) multichannel publishing§

Content analysis (already done earlier)

We can now plan our EDD and begin to build it.



Work with Structured Content¶

Adobe provides tools that work with structured (XML) content. These include both Adobe FrameMaker and XML Documentation for Adobe Experience Manager.¶

Adobe FrameMaker¶

Adobe FrameMaker is the industry-standard tool for authoring and publishing multilingual technical content across mobile, web, desktop, and print. Easily work with unstructured and structured content in the same documentation. Work faster and smarter with advanced XML/DITA capabilities.¶

Publish content as:¶

- Responsive HTML5¶
- Mobile App¶
- PDF¶
- EPUB¶
- and more¶

XML Documentation for Adobe Experience Manager¶

Dynamically deliver DITA content to *Adobe Experience Manager* thereby bringing marketing and technical content to the same platform. Effectively manage all aspects of your content workflow such as:¶

- authoring¶
- web-based review and collaboration¶
- translation¶
- digital asset management, and¶
- multichannel publishing§

The EDD will have support for many elements, but based on the review of the content, we know we have to include at least:

- topic, as a highest-level element. This is the overall document that we'll create.
- title, a required subordinate to the topic.
- section, an optional sub-level of content with its own title and content.
- paragraphs, usually at least one or more is required after a title.
- lists, optional and containing one or more items if lists are used.
- emphasized words, optional and appearing within paragraphs.

The great news is that in the earlier exercises we created a template and EDD that already support this content!

Unique tagging of content

For best results in conversion the content you work with should be correctly tagged. Uniquely tagged paragraphs in our sample include Title, Heading1, Bulleted, and so on. The same can be said about the character level format. If you have larger documents ensure they are based on well-defined templates and that the tagging is applied in a consistent way for best results. This allows us to convert unstructured content to structured content using a conversion table. Note that only tagging in use in the document will be reflected in a conversion table.

Migrating unstructured files to structure

To best manage the conversion of unstructured content you need:

- Content that is unstructured, but uses tags consistently.
- An EDD that has the formal rules you want to follow.
- A conversion table that maps content between unstructured and structured models.

We already have the first two. The content that we just created is unstructured. The EDD we created in the last hands-on portion has the rules. We still need a conversion table.

The conversion table is a mapping utility that makes it easier to transfer unstructured to structured content. Results will vary based on many factors. These can include the consistency of your source content, how close your source structured and formal rules for structure align, the use of some content types (for example, whether images are copied or referenced), and other specific issues that may come up.

The good news is that if you do anything consistently there is often a way to manipulate FrameMaker either via scripting before or after the transformation to structured content. While we'll just convert one document in this set of exercises it is possible to convert dozens, hundreds, or even thousands by using conversion tables and working with good quality source content.

Import EDD rules

To ensure the resulting conversion matches the structure, we need to first import the EDD into the current unstructured document.

1. Open *TopicEDD.fm*.
2. Switch to the *WorkWithStructure.fm* file.
3. Select **File** > **Import** > **Element Definitions** and, selecting the TopicEDD.fm, click **Import**. The element definitions are imported successfully. While the rules are imported the current unstructured document looks the same. However, if the Elements pod is displayed you can see elements to choose from.

Create a default conversion table

FrameMaker can automatically build a basic conversion table for you. In doing so the content is scanned and paragraph, character, and other tags (for example, tables, cross-references, markers, and so on) are found and mapped. The default mapping is to match the name of the tag with the name of an element.

1. Select **Structure** > **Utilities** > **Generate Conversion Table**.
2. Select **Generate New Conversion Table**.
3. Click **Generate** and review the table.

Wrap this object or objects	In this element	With this qualifier
P:Title	Title	
P:Body	Body	
P:Heading1	Heading1	
P:Bulleted	Bulleted	
C:Emphasis	Emphasis	

4. Save this in the same folder as your other files and name it ConversionTable.fm

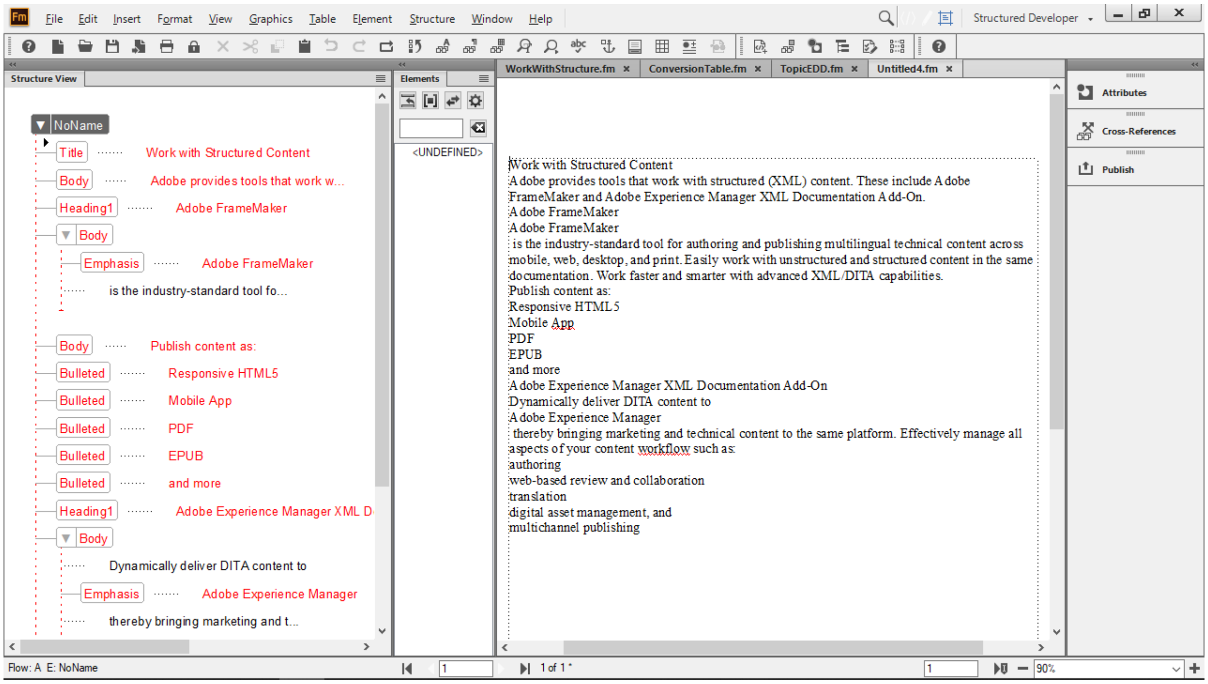
The resulting conversion table has a one-to-one mapping of the paragraphs and the character tag. While the resulting elements are not matches with the EDD we can still apply the conversion table, view the results, and start to make adjustments to the conversion table. After each change we can reconstruct the structured file and review it for accuracy.

Convert a document to structure

It is likely, or even certain, that the first conversion table will not give us the results we need. However, to see how the conversion of iterative updates impacts the results let us convert the WordWithStructure.fm file to a structured document using the conversion table default.

1. Switch to the *WorkWithStructure.fm* file.
2. Select **Structure** > **Utilities** > **Structure Current Document**.
3. Ensure the conversion table to use is the ConversionTable.fm file.
4. Click **Add Structure**.
5. When the operation is successfully completed click **OK**.

   The format is stripped of format and a default structure (that is incorrect) is applied.



Work with Structured Content
Adobe provides tools that work with structured (XML) content. These include Adobe FrameMaker and Adobe Experience Manager XML Documentation Add-On.
Adobe FrameMaker
Adobe FrameMaker
is the industry-standard tool for authoring and publishing multilingual technical content across mobile, web, desktop, and print. Easily work with unstructured and structured content in the same documentation. Work faster and smarter with advanced XML/DITA capabilities.
Publish content as:
Responsive HTML5
Mobile App
PDF
EPUB
and more
Adobe Experience Manager XML Documentation Add-On
Dynamically deliver DITA content to
Adobe Experience Manager
thereby bringing marketing and technical content to the same platform. Effectively manage all aspects of your content workflow such as:
authoring
web-based review and collaboration
translation
digital asset management, and
multichannel publishing

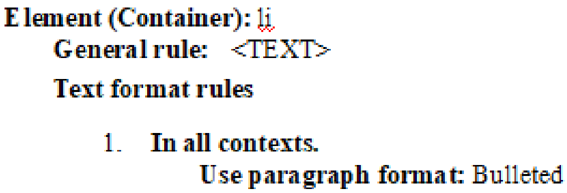
   The red in the structure indicates an element that does not match the rules. In this sample the highest level element is NoName (and in the EDD it should have been topic). Tags like Title, Body, Heading1, and Emphasis are direct conversions of the named tags.

6. Close the Untitled file.
7. Switch back to the ConversionTable.

Update the default conversion table

As you change a conversion table, the conversion process needs to be repeated to see the impact of the change.

As you update the structure the content adheres to the EDD rules and formatting is also applied. Remember, our EDD was built with a rule that formats the li as bulleted!

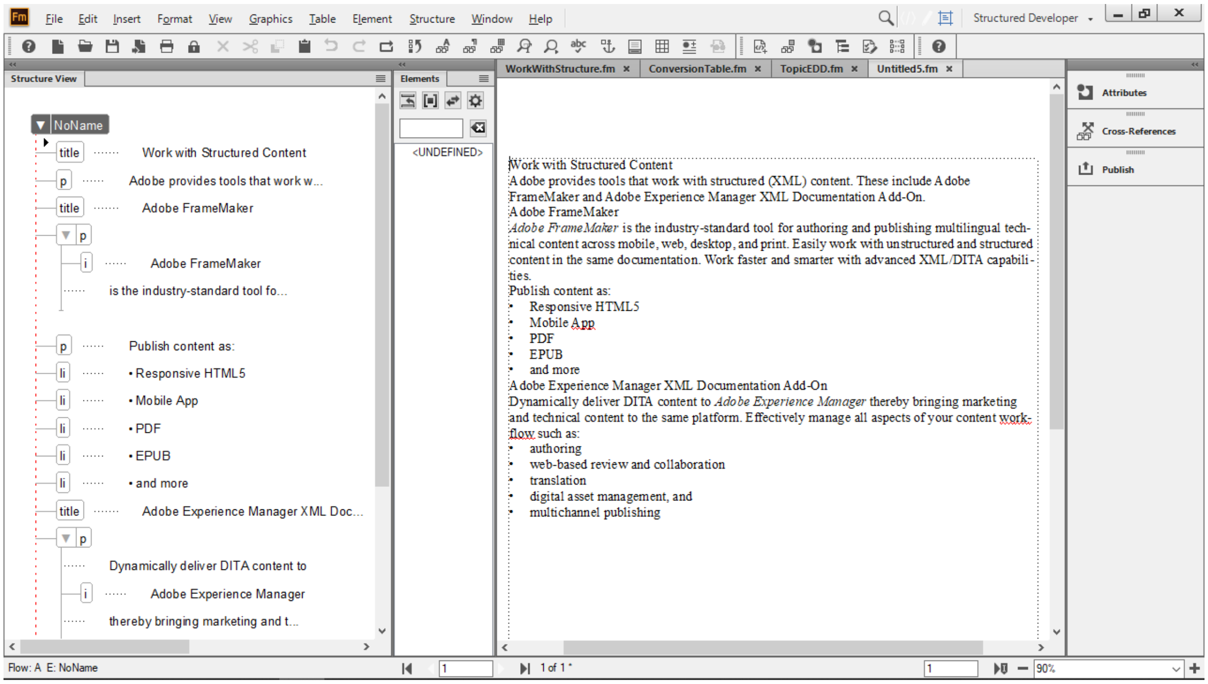


Element (Container): li
General rule: <TEXT>
Text format rules
1. **In all contexts.**
   Use paragraph format: Bulleted

1. Modify the conversion table by changing the second column.

Wrap this object or objects	In this element	With this qualifier
P:Title	title	
P:Body	p	
P:Heading1	title	
P:Bulleted	li	
C:Emphasis	i	

2. Switch to the WorkWithStructure.fm file.
3. Select **Structure** > **Utilities** > **Structure Current Document**.
4. Ensure the ConversionTable document is being referenced.
5. Add the structure and review the results.



Work with Structured Content

Adobe provides tools that work with structured (XML) content. These include Adobe FrameMaker and Adobe Experience Manager XML Documentation Add-On.

Adobe FrameMaker

Adobe FrameMaker is the industry-standard tool for authoring and publishing multilingual technical content across mobile, web, desktop, and print. Easily work with unstructured and structured content in the same documentation. Work faster and smarter with advanced XML/DITA capabilities.

Publish content as:

- Responsive HTML5
- Mobile App
- PDF
- EPUB
- and more

Adobe Experience Manager XML Documentation Add-On

Dynamically deliver DITA content to *Adobe Experience Manager* thereby bringing marketing and technical content to the same platform. Effectively manage all aspects of your content workflow such as:

- authoring
- web-based review and collaboration
- translation
- digital asset management, and
- multichannel publishing

The broken line in the structure indicates elements that are not in the expected order, based on the parent. In this sample the highest level element is NoName (and in the EDD it should have been topic). Tags are, however, much better. The title, p, i, and li elements are no longer in red. This is because they are valid as elements, but not structured as expected. In addition, the format has been assigned in the document view to the li elements. We now have better, but still incomplete converted content.

6. Close the Untitled file.
7. Switch back to the ConversionTable.

Nest list content

Rather than just one level of conversion a nested process can be completed. In our example we'll group the list items into a list.

1. Modify the conversion table as seen here by adding a new row at the end.

Wrap this object or objects	In this element	With this qualifier
P:Title	title	
P:Body	p	
P:Heading1	title	
P:Bulleted	li	
C:Emphasis	i	
E:li+	ul	

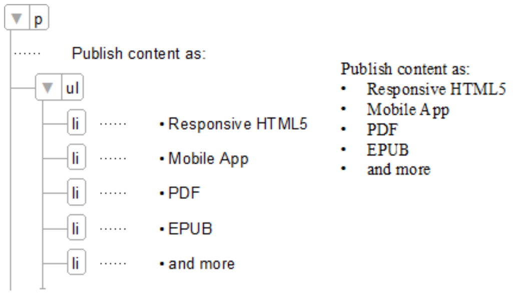
The rule we've added will take one or more **li** elements and wrap them in the **ul** element.

2. Switch to the WorkWithStructure.fm file.
3. Select **Structure** > **Utilities** > **Structure Current Document**.
4. Ensure the ConversionTable document is being referenced.
5. Add the structure and review the results.

   While the format remains the same, the structure is updated to show the **li** elements as children of the **ul** parent.
6. Close the Untitled file.
7. Switch back to the ConversionTable.

Nest lists in paragraphs

Nesting content ensures that if the parent element is selected (for example, to move it, delete it, or to reuse it) the nested content is also included. For example, if a paragraph introduces an idea (such as the text string "Publish content as:") then the following text should be considered a part of the same element.



In this example if the **p** element is selected the **ul** (and each of its child **li** elements) is also selected.

1. Change the rule for the conversion of the paragraph named Body to support an optional subordinate **ul** element.

Wrap this object or objects	In this element	With this qualifier
P:Title	title	
P:Body, **E:ul?**	p	
P:Heading1	title	
P:Bulleted	li	
C:Emphasis	i	
E:li+	ul	
E		

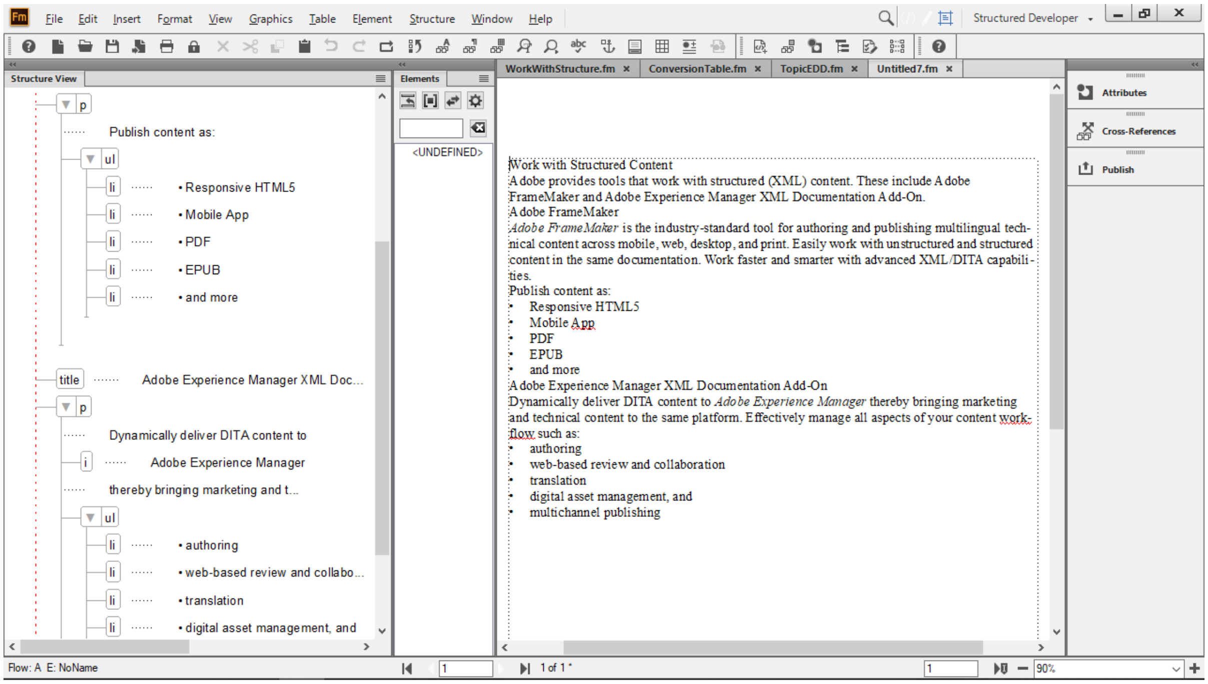
The new rule is going to find the paragraph named Body and, if followed by an optional element named **ul**, convert it into a **p** element. If the Body is not followed by the **ul** element it will still convert. The question mark indicates "zero or one".

2. Switch to the WorkWithStructure.fm file.

3. Select **Structure** > **Utilities** > **Structure Current Document**.

4. Ensure the ConversionTable document is being referenced.

5. Add the structure and review the results.



Work with Structured Content
Adobe provides tools that work with structured (XML) content. These include Adobe FrameMaker and Adobe Experience Manager XML Documentation Add-On.
Adobe FrameMaker
Adobe FrameMaker is the industry-standard tool for authoring and publishing multilingual technical content across mobile, web, desktop, and print. Easily work with unstructured and structured content in the same documentation. Work faster and smarter with advanced XML/DITA capabilities.
Publish content as:

- Responsive HTML5
- Mobile App
- PDF
- EPUB
- and more

Adobe Experience Manager XML Documentation Add-On
Dynamically deliver DITA content to *Adobe Experience Manager* thereby bringing marketing and technical content to the same platform. Effectively manage all aspects of your content workflow such as:

- authoring
- web-based review and collaboration
- translation
- digital asset management, and
- multichannel publishing

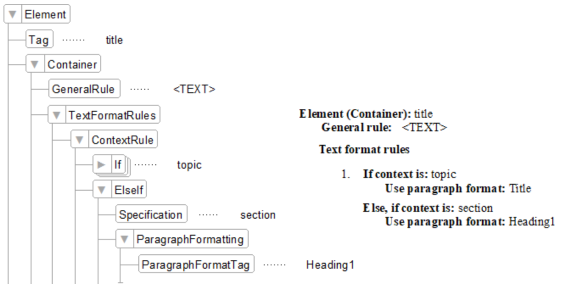
The structure is updated and shows the **ul** elements as children of the **p** parent.

6. Close the Untitled file.

7. Switch back to the ConversionTable.

Convert titles using qualifiers

Remember that our EDD defined rules for a title that provided a format that changed based on context.



Element (Container): title
General rule: <TEXT>

Text format rules

1. **If context is:** topic
   Use paragraph format: Title

   Else, if context is: section
   Use paragraph format: Heading1

A **title** in the **topic** element is formatted as a Title paragraph and, if in the **section,** it is formatted as a Heading1. Currently we are mapping both the Title and Heading1 to a title. By using a qualifier we can treat the two in different ways. The Title paragraph will become the "main" one and the Heading1 will become the "sub". This allows us to convert and group the content in unique ways to create **section** or **topic** elements.

1. Change the rule for the conversion of the paragraph named Title and the one named Heading1 to distinguish them.

2. Add a new row to the table to structure content into a **section** element.

Wrap this object or objects	In this element	With this qualifier
P:Title	title	main
P:Body, E:ul?	p	
P:Heading1	title	sub
P:Bulleted	li	
C:Emphasis	i	
E:li+	ul	
E:title[sub], E:p+	section	

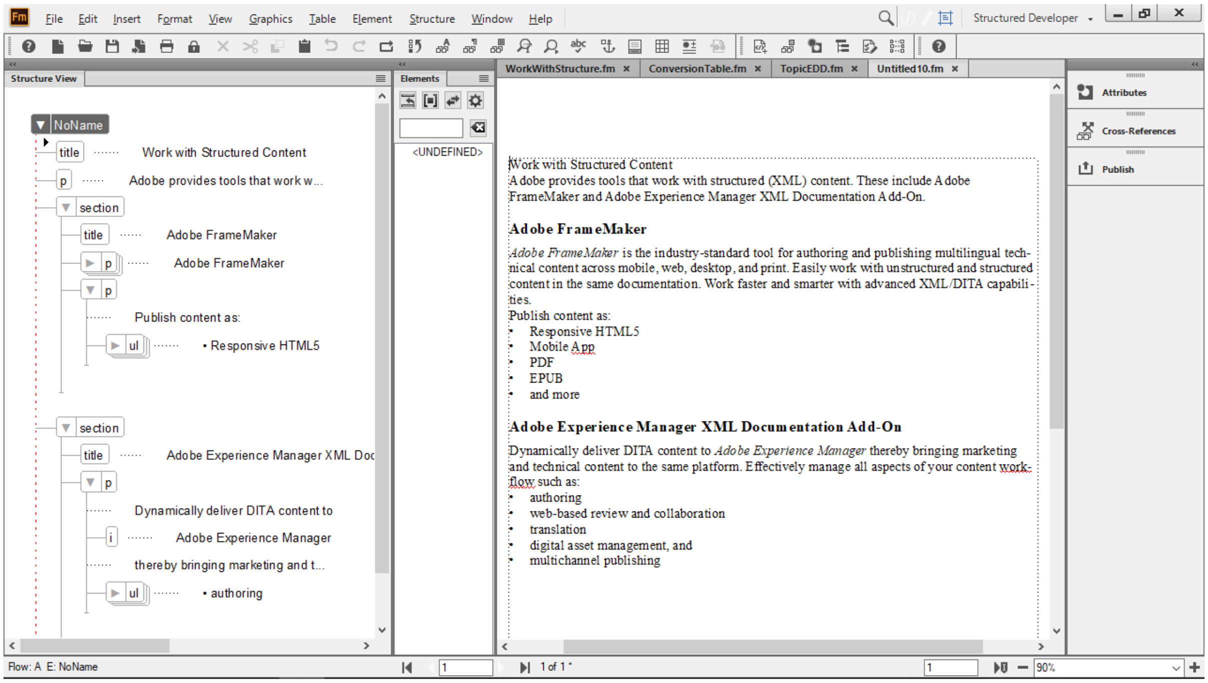
The rules provide a distinct qualifier for the two **title** elements. The new row will find the element named **title**—and if it is marked with the "sub" qualifier AND followed by one or more **p** elements (the plus sign means "one or more")—and wrap these into a **section** element.

3. Switch to the WorkWithStructure.fm file.

4. Select **Structure** > **Utilities** > **Structure Current Document**.

5. Ensure the ConversionTable document is being referenced.

6. Add the structure and review the results.



Work with Structured Content
Adobe provides tools that work with structured (XML) content. These include Adobe FrameMaker and Adobe Experience Manager XML Documentation Add-On.

Adobe FrameMaker

Adobe FrameMaker is the industry-standard tool for authoring and publishing multilingual technical content across mobile, web, desktop, and print. Easily work with unstructured and structured content in the same documentation. Work faster and smarter with advanced XML/DITA capabilities.
Publish content as:
- Responsive HTML5
- Mobile App
- PDF
- EPUB
- and more

Adobe Experience Manager XML Documentation Add-On

Dynamically deliver DITA content to *Adobe Experience Manager* thereby bringing marketing and technical content to the same platform. Effectively manage all aspects of your content workflow such as:
- authoring
- web-based review and collaboration
- translation
- digital asset management, and
- multichannel publishing

The structure is updated and two **section** elements are created. Each contains a **title** (properly formatted) and one or more **p** elements.

7. Close the Untitled file.

8. Switch back to the ConversionTable.

Complete the topic conversion

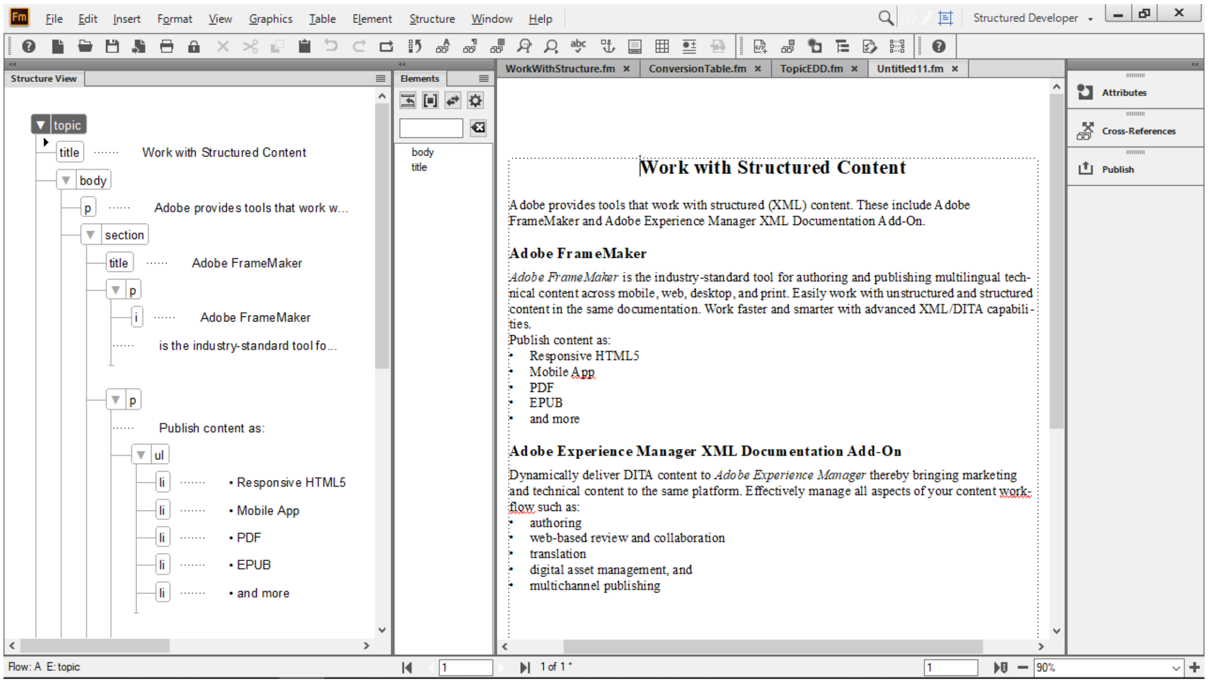
A few more modifications using the ideas we've learned allows us to create a complete conversion of the topic.

1. Update the conversion table to the following.

Wrap this object or objects	In this element	With this qualifier
P:Title	title	main
P:Body, E:ul?	p	
P:Heading1	title	sub
P:Bulleted	li	
C:Emphasis	i	
E:li+	ul	
E:title[sub], E:p+	section	
E:p, E:section+	body	
E:title[main], body	topic	

The rules provide a distinct qualifier for the two **title** elements. The new row will find the element named **title**—and if it is marked with the "sub" qualifier AND followed by one or more **p** elements (the plus sign means "one or more")—and wrap these into a **section** element.

2. Switch to the WorkWithStructure.fm file.
3. Select **Structure** > **Utilities** > **Structure Current Document**.
4. Ensure the ConversionTable document is being referenced.
5. Add the structure and review the results.



Work with Structured Content

Adobe provides tools that work with structured (XML) content. These include Adobe FrameMaker and Adobe Experience Manager XML Documentation Add-On.

Adobe FrameMaker

Adobe FrameMaker is the industry-standard tool for authoring and publishing multilingual technical content across mobile, web, desktop, and print. Easily work with unstructured and structured content in the same documentation. Work faster and smarter with advanced XML/DITA capabilities.

Publish content as:

- Responsive HTML5
- Mobile App
- PDF
- EPUB
- and more

Adobe Experience Manager XML Documentation Add-On

Dynamically deliver DITA content to *Adobe Experience Manager* thereby bringing marketing and technical content to the same platform. Effectively manage all aspects of your content workflow such as:

- authoring
- web-based review and collaboration
- translation
- digital asset management, and
- multichannel publishing

The structure is updated and two **section** elements are created. Each contains a **title** (properly formatted) and one or more **p** elements.

6. Save the Untitled file as ValidTopic.fm in the same location as your other files.
7. Switch back to the ConversionTable.
8. Save and close the ConversionTable.

9. Save and close all files EXCEPT for the ValidTopic.fm.

Compare the valid topic with DITA

A lot of work has been done to get to this point. Comparing the results with a valid DITA topic will show you how close the converted content is to a valid topic in the DITA model! Since the default templates that are included have different formats you will see a few other benefits of working with structured content. For example, as soon as we transfer this to a new template the format will change to reflect

1. In the ValidTopic file click the topic element and copy it (and all the content in it).
2. Select **File** > **New** > **DITA Topic**.
3. In the untitled DITA topic, in the Structure view, click in the topic element.
4. Delete the element and all the default content.
5. Select Edit > Paste.

   The content from your topic is pasted into the shell of the DITA topic and can be reviewed for accuracy.

   By default the template may display all attributes; this can be toggled to show required or specified attributes.
6. Select View > Attribute Display Options.
7. Set the display to show **Required and Specified Attributes**.
8. Scroll to the top of the Structure view.
9. Review the DITA topic.
10. The only invalid part of the conversion is the required ID attribute.
11. Right click in the topic element and select **Assign ID to Element**.

Tips and tricks about conversion to structure

Conversion tables are powerful tools. The sample provided here is just one small part of what can be done during conversion.

When mapping content, start with a document that you can manage the complexity of. Don't try to convert a document that is 5, 10, or 50 pages long. Instead, start with something that is representative, but only a page or two long. This may provide you a good way to learn without having to try to juggle too many rules at once. However, when you are ready, here are some things to think about to continue to develop you conversion tables.

Start simple

Begin by structuring the simpler elements first. That may be the character formats in the content, simple paragraphs, or basic lists. Create structure from the smaller parts to begin. Then wrap them as we did in the exercises.

Wrap this object or objects	In this element	With this qualifier
P:Title	title	
P:Body	p	
P:Heading1	title	
P:Heading2	title	
P:Bulleted	li	
P:Numbered1	li	

Wrap this object or objects	In this element	With this qualifier
P:Numbered	li	
C:Emphasis	i	
C:Heavy	b	
C:Menu	uicontrol	

Use qualifiers

Look for duplicate elements, similar to the two title elements we had and consider uniquely qualifying them. For example, you may have both numbered and bulleted content that become li elements. One set could be qualified as being 'numbered' and the other as 'bulleted' to ensure conversion into appropriate parent elements.

Wrap this object or objects	In this element	With this qualifier
P:Title	title	main
P:Body	p	
P:Heading1	title	sub
P:Heading2	title	section
P:Bulleted	li	bullet
P:Numbered1	li	number
P:Numbered	li	number

Add a root element

The root element is the "highest level" element. In our sample we converted content and eventually wrapped it all in the topic element. However, adding RE:RootElement and then naming it is another way to define the top level. Only one root element may be defined per conversion table.

Our sample wrapped content and resulted in a valid structure, but to get there took several passes. By using the RE for a root element you can quickly assign the top level structure every time instead of working around the NoName.

Wrap this object or objects	In this element	With this qualifier
E:title[main], body	topic	
RE:RootElement	topic	

Work iteratively

Make a change. Test it. Repeat. This allows you to make small changes and see what the impact is. By making incremental strides you can identify an error much quicker. Instead of making 4 or 8 or 25 changes, make 1 or 2. Toggle to the unstructured file and apply the conversion table. Repeat this as needed until you succeed. Then move ahead.

Save your file as needed, but you don't need to save to import. The most current content is always used, so you can test the conversion and, if the results are not as expected, you can use the Undo command to reset. If you plan to do complex changes you may want to save multiple versions of the content. Create your ConversionTable, but every so often create a backup (for example, you may end up with ConversionTable01, ConversionTable02, and so on) so that you can review the work you have done.

Add to the conversion table

Once you have a working conversion table for a smaller file, open a larger document. By again selecting **Structure** > **Utilities** > **General Conversion Table** you can select an existing conversion table and add to it by selecting the current conversion table. The new content is added at the end of the current table.

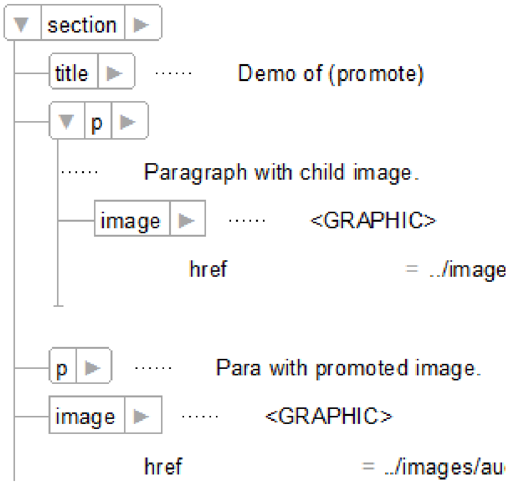
Create multiple conversion tables

Structures such as DITA have multiple root elements. For example, a topic, concept, task, reference, or glossary could be created. In each of these there may be subtle (or major) differences. The topic, concept, and reference have similar content within them (for example, each supports section and example elements, but they also have a unique body, conbody, and refbody. To structure them from the "bottom to the top" may mean creating more than one conversion table to support the migration to structure. Start with one, then save it with a new name and modify it as needed.

When we convert content for clients we ask them to identify content as a task, concept, reference, and so on in the source. Then we uniquely structure them using 3 or more conversion tables to quickly create the correct hierarchy of content.

Promote content (graphics and tables) if needed

A graphic or a table is often anchored into the preceding content and may need to break free. The default conversion means they often end up as children of the content they originally were anchored in. Using the promote command supports converting content to a sibling of the original paragraph rather than as a child.



In this sample the first **p** and **image** are the result of a default conversion, and the second the result of (promote) being used.

Wrap this object or objects	In this element	With this qualifier
G:Graphic	image(promote)	
T:Table	tgroup(promote)	

Add attributes

In some conversions attributes should be added. For example, if the element note has an attribute for type then the value may need to be defined for a tip, caution, warning, default note, and so on.

Wrap this object or objects	In this element	With this qualifier
P:Tip	note[type="tip"]	
P:Caution	note[type="caution"]	

Wrap this object or objects	In this element	With this qualifier
P:Warning	note[type="warning"]	
P:Note	note	

Handling formatting overrides

One common issue is the use of formatting overrides. Consider using the File > Utilities > Create and Apply Formats and then map each resulting tag to a specific row in a conversion table.

Other conversion challenges

There are many challenges related to working with source content and structuring it. You need to know the structure you want to convert to, be familiar with the content, and have an understanding of the EDD and templates.

Challenges may include:

- Large documents that must be converted to multiple document types. For example, a large chapter may need to be structured with content in a **task**, **concept**, and **reference**. A single conversion table to manage all this could be a challenge. The splitting of the document could also be difficult.
- Manual formatting is added to the documents and must, based on context of use, be uniquely converted. For example, the +Bold is applied (by pressing Ctrl+b, or even using proper character tags), but if it is in a step it should be a **uicontrol** element and if in a note it should become a **b** element. Converting all of this to one element is not ideal, but you cannot use a qualifier on the exact same tag to create two unique structures.
- Images are imported by copy and should be referenced instead. For example, someone created screenshots using a print screen utility, cropped it, and then copied and pasted it directly into content. Structuring the document creates a valid document, but on export to XML the image is discarded.
- Uniquely structuring tables based on the table format. For example, a table with complex content should be converted to a **tgroup**, but a very basic data table should be structured as a **simpletable**. This cannot be done by default.

The bad news is that there are many, many, many things users can introduce to your source content that cannot be addressed simply by using a conversion table. However, many of the issues addressed above (and other common and repeatable issues) can be resolved by automation using scripting.

Where to go from here

The content we created is comprehensive. There is a lot of information here; however, it's important to read additional materials as you start to make the move to structured authoring.

The move to structured authoring, template developing, formatting, and publishing can be challenging. Add to that the growing interest in managing content in a component content management system. To ensure a smooth transition from unstructured to structured content you may want to read more, visit the sites for specific standards, attend conferences, get trained, or hire a consultant.

- **Adobe content**
- **Standards**
- **Training**
- **Consulting**

Adobe content

Adobe supports the FrameMaker community with videos, tutorials, whitepapers, and much more. To find out more:

- **https://www.adobe.com/products/framemaker/whitepapers.html** has several whitepaper on FrameMaker.
- **https://www.adobe.com/products/framemaker/features-all.html** includes all the features of FrameMaker, including a "Getting Started" section.
- Documentation to dive in deeper with regards to structured content can also be downloaded in PDF format. At over 250 and 450 pages in length the developer's guide and reference manual are for anybody who develops structured FrameMaker templates and XML or SGML applications. They are not written for end users who author structured content.
- **https://help.adobe.com/en_US/framemaker/pdfs/Structure_Dev_Reference.pdf**
- **https://help.adobe.com/en_US/framemaker/pdfs/Structure_Dev_Guide.pdf**

Standards

Many of the leading XML standards have their own official sites where you can dive in much deeper. In most cases a quick search for standard results in discussion groups, conferences, online events, trainers, consultants, and other ways to learn more. As each standard is constantly changing it's best to do your own research to learn more about the solutions available by standards. Common standards that are supported in FrameMaker include:

- DITA
- DocBook
- S1000D

Training

Training is available from authorized training providers and by instructors who have years of practical and hands-on experience working with FrameMaker. Additional resources can also be found on the Adobe website.

Publishing Smarter delivers FrameMaker and structured FrameMaker training and has done so for over 20 years. Training is available online or in-person and covers topics including using FrameMaker as a structured (or unstructured) author, template developer, or even as the developer of an EDD and related documents. Visit **www.publishingsmarter.com** for more details.

Consulting

Many consultants can be found who bring their experience to you. They can provide guidance and help you troubleshoot, work with you to set up a complete structured workflow, or even manage the entire project for you.

Publishing Smarter delivers FrameMaker and structured FrameMaker consulting and has done so for over 20 years. We know the ins and outs of structured content and templates. We've been luck to work with Adobe since they first acquired FrameMaker. We've helped them build templates that are included out-of-the-box including structured and unstructured templates. We created sample content, helped Adobe build the AEM and DITA tools, and actively continue to work with the FrameMaker development team. We know the templates, the people, the workflows, and the challenges that can come with a migration to structure. We've also spent years developing processes to automate the conversion of legacy content, built hundreds of templates for use with DITA, converted thousands of documents to structure, and can bring that experience to your projects.

Consulting is available online or in-person. Visit **www.publishingsmarter.com** for more details.

More information

https://www.adobe.com/products/framemaker.html is home to the FrameMaker product page. From there you can read much more about the product, watch educational videos, and explore structured content.

About the author

Bernard Aschwanden solves documentation-based problems and helps companies generate more revenue. He guides clients through best processes to create, manage, and deliver content. Once content is delivered, he helps socialize the message, understand and act on feedback, and improve the process and workflow. He is the founder of Publishing Smarter (**www.publishingsmarter.com**), an Associate Fellow of STC, and a Past President of STC. Bernard has helped hundreds of companies implement successful solutions. He is focused on publishing better, publishing faster, and publishing smarter.

Get in touch

✉ techcomm@adobe.com ☎ +1-866-647-1213

Adobe

Adobe Systems Incorporated
345 Park Avenue
San Jose, CA 95110-2704
USA
www.adobe.com

Adobe, the Adobe logo, and FrameMaker are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. All other trademarks are the property of their respective owners.

© 2022 Adobe Inc. All rights reserved.