

# *THE HITCHHIKER'S GUIDE TO XML AUTHORIZING*



## Contents

<b>Does My Organization Need XML?</b>	<b>3</b>
<b>Structure and XML—The Business Case</b>	<b>3</b>
<b>Preparing For the Journey—Understanding Structure and XML</b>	<b>5</b>
• Introducing Structured FrameMaker	5
• Introducing XML	7
<b>Why Use XML?</b>	<b>7</b>
• XML is Open, Not Proprietary	8
• XML Supports Standards Like DITA, DocBook, or S1000D	8
• XML Has Rules	8
• Database Publishing—XML As Data Transport	9
• XML Supports Reuse	9
• XML Integrates With Localization Processes	10
• XML Supports Collaboration	10
<b>Why Use Structured FrameMaker?</b>	<b>10</b>
• Authoring Visually	10
• Editing XML Code	11
• Excellent Print and PDF Output	12
• Automatic and Enforced Formatting	13
• On-The-Fly Validation (Guided Editing)	13
• Relatively Low Licensing Cost	14
<b>Why Use FrameMaker XML Author?</b>	<b>14</b>
<b>Components of a Structured Authoring Environment</b>	<b>15</b>
• Authoring Only in Structured FrameMaker	15
• Structured Authoring and XML	16
<b>Using FrameMaker Structure and XML</b>	<b>17</b>
• Content Analysis	17
• Building the Proposal EDD	18
• Adding Formatting to the EDD	23
• Building a Structured Application	26
<b>Migrating Unstructured Files to Structure</b>	<b>29</b>
• Cleaning Up Legacy Documents	30
• Creating the Conversion Table	30
• Conversion Example	32
<b>Conclusion</b>	<b>36</b>
<b>Where To Go From Here</b>	<b>37</b>
• Adobe White Papers	37
• Recorded Adobe Webinars	37
• Docbook	37
• Training	38
• Consulting	38

The all-new Adobe  
FrameMaker (2019  
release)



Adobe FrameMaker (2019  
release)

For a 30 day free trial of  
FrameMaker, visit

[www.adobe.com/go/  
tryframemaker](http://www.adobe.com/go/tryframemaker)

To schedule a private  
demo, visit

[www.adobetechcommdemo.com](http://www.adobetechcommdemo.com)

# The Hitchhiker's Guide to XML Authoring

If you're a technical publications professional, you've undoubtedly heard talk about structured documentation, XML, and Structured Adobe FrameMaker. You may be thinking about these technologies, and wondering how you can put them to use.

This guide will answer questions you might have, and give you information about:

- How structured and unstructured documents compare
- The benefits of structure and XML
- How Structured FrameMaker fits in
- Structured authoring environments
- Converting unstructured documents
- Where to go from here

## Does My Organization Need XML?

This is the first question to ask before embarking on setting up an XML authoring system. Or at the very least, you should ask whether your organization could get value from using XML. There is no question that there is a cost to setting up an XML system. Before making the investment, you should be sure that it will benefit your organization, both within itself, and through the sharing of your assets with other organizations.

Some questions you can ask include:

- Is our authored content an important business asset?
- Must we comply with existing standards (S1000D, Docbook, DITA, etc.)?
- Is it important to share information assets with other organizations?
- Is collaborative authoring important?
- Do we need to support different authoring/management tools to operate on the same content?
- Is it important to merge authored content with other information assets?
- Do we deliver the same information in multiple output formats?
- Do we need to mix and match content in different publications?
- Are we trying to improve or personalize the customer experience?
- Are we interested in "intelligent content" or "adaptive content"?
- Do we localize our content?
- Do we need to manage our content in source control, or in even more powerful content management systems?

It's difficult to say how many "Yes" answers mean your organization needs to manage its content in XML. But the more you can answer "Yes" to these questions, the more you should consider XML as an enabling technology that will have a good ROI for your organization, and for your internal and external customers.

## Structure and XML—The Business Case

Before embarking on a conversion from unstructured documentation to structured and XML, you should have an idea of the value it will add to your business. Information is an increasingly important business asset.



The ability to manage and process assets is always an important part of any business. It follows that improved management and processing of information adds value to your information assets.

To answer modern information requirements, we see new positions within the enterprise, including:

- Information Architecture
- Information Design
- Information Engineering

These positions have one thing in common; they focus on how computer processing can improve the delivery, management, and creation of information assets. One other thing they have in common is this: Selective Feature Capability Comparison

1. You can publish to multiple formats automatically. Outcomes: With the same content, you can easily produce PDF, online, and mobile output. You can set up a single build process that publishes multiple formats at once. Responsive content.
2. Your content becomes more consistent, lowering costs and increasing flexibility. Outcomes: Templated information such as knowledge-base articles, maintenance procedures, or proposals can be automatically verified for completeness. Tools can more easily process consistent data.
3. Your authors can get back to what they do best—creating quality content. Outcomes: Less time spent on formatting decisions. Better division of labor in the organization.
4. Your content becomes more predictable, therefore more usable. Outcomes: Users know what to expect and where to look in the information product. Increased usability increases user base and revenue.
5. You can integrate information flows for more accurate, complete content and better products. Outcomes: Breakdown of information silos. Automated processes to integrate information from different departments.
6. You can automate and improve your business processes. Outcomes: Documentation data integrates with other data processing. Automated transport of data between departments and processes. Provide a single information flow to suppliers and customers. Reduce localization costs—integrate with localization processes.
7. Your content becomes intelligent and adaptive, supporting more targeted marketing. Outcomes: The right information for the right user. Content adapts to reader's role and context.
8. You can use a content management system for unprecedented efficiencies. Outcomes: Manage inventory of information assets. Track information changes and versions. Manage references and other dependencies automatically. Assemble document deliverables from a pool of topics. Reuse content.
9. You can personalize your content, increasing response rates. Outcomes: Increase consumer interaction by responding to preferences, roles, and context. Include user-authored content (comments, tips, ratings, etc.).
10. You can reuse your content, lowering costs while improving quality. Outcomes: Write topic once, use often. Reduce localization costs—translate once per language, use often.

For further reading, you can see the Adobe blog post, "Ten Reasons to Structure Your Content" at <http://blogs.adobe.com/techcomm/2013/09/white-paper-ten-reasons-to-structure-your-content.html>. This includes a link to the full white paper of the same name.

## Preparing For The Journey—Understanding Structure and XML

Before you start working with structured documents, you need to get ready. A good start is to understand what structure is all about, and the value you can get out of incorporating structure and XML in your work flow.

If you are a FrameMaker user then you should already be familiar with the idea of structuring your writing. You adhere to your template design, and you use headings, lists, and other formatting to categorize and organize the ideas in your work. In other words, you give your work a conceptual structure. As a user of unstructured FrameMaker you capture that structure by using the template consistently.

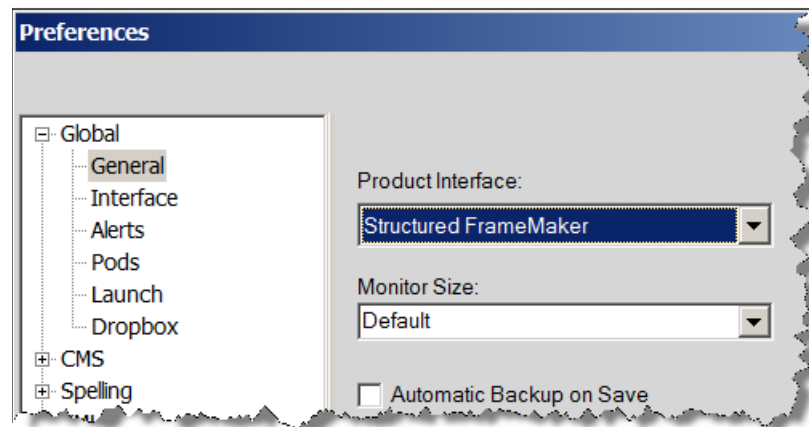
Using the template correctly not only helps you write in a structured way, but it also makes it easier for FrameMaker to process the document. Consistent use of paragraph formats sets up tables of contents and lists of figures, and paragraph formats can be used to automatically apply master pages. FrameMaker also uses the document's structure when saving a document as HTML, breaking the document into separate HTML files, or extracting content to post in header or footer areas of each page.

For all the ways FrameMaker can process unstructured documents, imagine how much more processing you could perform if you captured more information about the document. Structure is a way to capture extra information about a document so that a computer can process the document in more ways.

### Introducing Structured FrameMaker

For a FrameMaker user, Structured FrameMaker is the easiest way to experiment with structured documents. It comes with ready-made templates that illustrate the extra value structure gives you. If you currently use unstructured FrameMaker, just open the Preferences dialog box and set the Product Interface to Structured FrameMaker. This won't change any of your existing work, and you can still use all the unstructured features you know. The Structured FrameMaker interface simply enables extra features that work with structure.

Setting the user interface to Structured FrameMaker

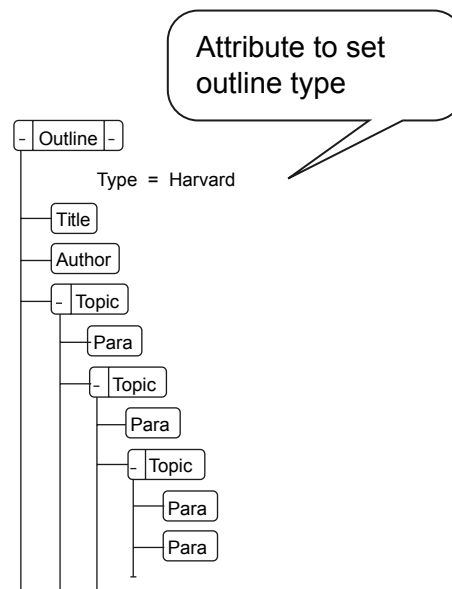


When you run Structured FrameMaker, you can open samples of structured templates and view the structure in the user interface. To open a structured template, choose *File > New > Document*—in the dialog box, click Explore Structured Templates to open the structured template browser. From there you can open samples of many different structured templates.

A structured template includes a definition of the structure a document can have. This includes the names of the different structure elements, rules for what an element can contain, attributes that describe various aspects of the elements, and rules to automatically apply formatting to elements depending on where they are in the document.

Structured FrameMaker is like an outliner running on hyper-drive. In fact, the structured template browser includes outline samples you can open and explore. The Structure View shows the outline topics in a hierarchical tree that corresponds with the outline's hierarchy (choose *StructureTools>Structure View*).

### Structure View for an Outline



In the Structured View, you can drag topic bubbles to different locations to promote or demote a topic, or to move a sub-topic into a different parent topic. Each bubble indicates a structure element—elements contain document content or other structure elements. When you drag an element to another location, FrameMaker updates the formatting.

In addition to hierarchy, structure assigns attributes to elements. In the outline sample, you can see a Type attribute (click the right-hand PLUS mark in the bubble). Double-click the attribute to choose among preset outline types, and FrameMaker automatically changes the outline format. This shows more of FrameMaker's formatting power—it can set document formatting according to element attributes and document structure.

The list of sample templates includes more complex documents such as reports or multi-chapter books. You should explore these samples to see more of the things you can do with structured documents.

This is a very brief introduction to structure in FrameMaker. The important points to understand are:

- A structured template can define the hierarchy a document will have
- The structure view displays document hierarchy as a tree of elements
- You can reorganize entire blocks of the document by dragging elements within the Structure View
- The template includes formatting that automatically adjusts according to element position
- Elements can have attributes that affect formatting or other processing of the document

But the most important point is this—because a document includes structure as metadata, or information about the document, this gives FrameMaker more opportunities to automatically process the document. Element position can drive formatting of the text. Element attributes can drive automatic processing such as formatting, conditional text (or filtering), header/footer display, links, or even custom implementations that perform document processing. The information in structure simply provides that many more hooks a computer can use to automatically process the document. Automation means cost savings, fewer mistakes, and less time when creating and maintaining most types of technical documentation.

## Introducing XML

The description of structure above is focused on Structured FrameMaker, one specific implementation that uses the added value of structure. Extensible Markup Language (XML) is a vendor-neutral, open format managed by the World Wide Web Consortium (W3C). It also expresses the structure of a document, but through an open standard. This means two important things:

- Anybody can implement a process that complies with the XML standard
- Any processes that comply with the standard can share document data

If you've looked at HTML source, XML will seem familiar to you. Like HTML, XML consists of tags enclosed in angle brackets (such as <example>), and is stored in a text file. The tags declare specific elements, the same as the bubbles in the FrameMaker Structure View show the document elements.

Just as a FrameMaker structured template includes information to declare the elements a document can have, you can declare the set of elements for an XML document type, including rules for what the elements can contain, and attributes for the elements. In fact, XML is very much like Structured FrameMaker, but with two major differences:

- XML is an open standard that uses text files to store document data
- Unlike Structured FrameMaker documents, XML files do not include formatting

An XML file might look like the following:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE concept PUBLIC "-//OASIS//DTD DITA Concept//EN"
    "../dtd/technicalContent/dtd/concept.dtd">
<concept id="conceptconcept" xml:lang="en-us">
  <title>Garage Concepts</title>
  <shortdesc>A well-stocked garage can be the envy
of the neighborhood.</shortdesc>
  <conbody></conbody>
</concept>
```

You can see that the document file is made up of text characters. This has two major advantages:

First, you can open the file in any text editor, and you can modify the file with any text editing process. The file is said to use an open standard, and anybody can create software to edit XML files. Not only can a company sell authoring software, but programmers can implement tools to modify the document in different ways. Thanks to this openness, you have probably loaded many XML files in your web browser or other software without even knowing it.

On the other hand, because the file is in plain text, but it includes elements, attributes, and other data, it is very difficult to work with raw XML. The above example is very small, but even that is hard to read. Imagine a full technical manual for a sewage treatment plant in raw XML. Reading and authoring so much XML isn't practical without the help of software to format the documents and manage the metadata.

## Why Use XML?

If you made it this far, then you have an idea about what a structured document is, and you know that XML is yet another format to capture document data—structure and content. But you should pause to understand what XML can do for you, and be clear about why you want to embark on migrating from unstructured FrameMaker documents to Structured FrameMaker and XML.

## XML is Open, Not Proprietary

From the start, a primary goal of XML has been to promote interoperability—sharing of data between different processes. This means that you can create a document with one software application, and open it or process it in another. Many organizations choose to use XML because it frees them from relying on one specific application.

The openness of XML also makes it easier to create extra tools to process documents. The most commonly used open source tool is the DITA Open Toolkit, which converts DITA XML to different formats, including CHM, PDF, WebHelp, and Eclipse Help. The point is, the XML community is large, and it has many contributors.

## XML Supports Standards Like DITA, DocBook, or S1000D

There are a number of existing XML standards that focus on documentation for specific industries. For example, the S1000D standard was developed for military aircraft, and has since been expanded to support documentation for land, sea, and commercial equipment.

For each standard, there is a robust community that can offer wisdom via forum posts, templates, open source tools, and products that focus on getting the most out of these standards. Many industries expect documentation to support these standards, and you might be planning to migrate your documentation into compliance.

## XML Has Rules

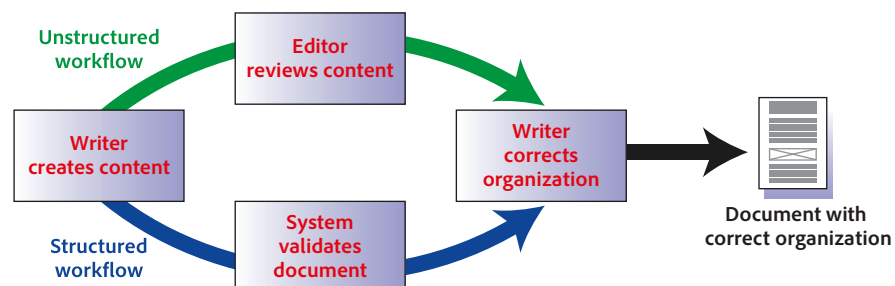
XML includes rules that describe the required organization of each element in the document. An XML file that follows these rules is valid, and you can use software to automatically make sure a document or book is valid.

For example, you might specify that a simple proposal should have the following content sequence:

1. Proposal title
2. Executive summary
3. Project description
4. Schedule
5. Cost

With unstructured documents you can suggest this structure, but you can't enforce it. With XML the content rules can be enforced, and a parser can verify that the document is valid. For documentation that must comply with standards, this can greatly reduce errors and cost.

## Human vs. Automatic Validation of Structure

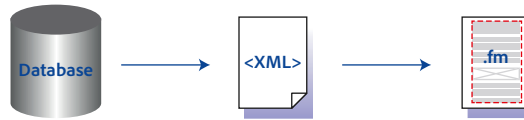




## Database Publishing—XML As Data Transport

XML is increasingly put to work as a way to send data from one application to another. Using structure to store content in elements translates easily into a database structure. Most databases can export data as XML which can then be brought into a document. XML is a perfect medium for database publishing. For example, you can use XML as an intermediary to publish a database in Structured FrameMaker.

Database Publishing with FrameMaker



## XML Supports Reuse

There are a number of ways to reuse XML documents. The bottom line is that you can write something once, and use it often. We all know about the use of boilerplate content. Re-use takes that idea to a higher level.

### Same Topic/Multiple Documents

XML standards support topic-based writing. XML makes it easy to build and manage multiple documents that share the same topic. For example, if you have many products that include the same login procedure, you only write that once, and use it for every product.

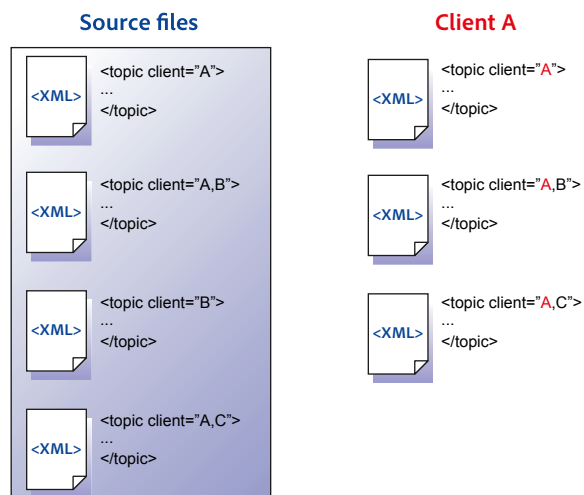
### Filtered Content

By using attributes and other metadata in XML, you can fine-tune your output. For example, DITA includes features to filter content based on audience or platform, and ways to manage cross-references or other referenced content for different outputs. For printed output a document can include the full content, but for mobile output you might choose to limit it to short descriptions. Because XML describes structure, this is easy to do.

Or consider the problem of customized software development. If your company customizes its products for each client, you need a way to publish different versions of the same document. Using metadata, you can specify which clients each topic applies to. When you are ready to deliver the information, you filter based on the metadata to create the appropriate output.

Filtering by attribute for client-specific deliverables

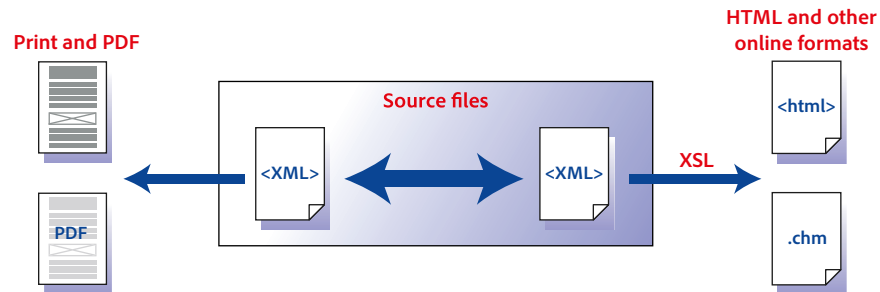
### Multiple Outputs



You can use the same content to generate documents for print and online use. Tooling exists for XML standards like DITA and Docbook to publish the same content to many outputs. If existing tools don't give the output you need, you can use open source technologies to implement custom outputs. From the same source, you can create outputs for print, desktop, or mobile devices, and use filtering to fine-tune the results.

Generating different outputs from XML source

### XML Integrates with Localization Processes



Content localization can represent a large part of publishing costs. Exact numbers vary for product localization, but as a general rule about a third of the total localization cost is for publishing and production. A publishing work flow 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.

In addition, most localization vendors use tools that integrate with XML. Especially for publications that use XML standards such as DITA or S1000D, the tooling is in place to streamline localization projects. When you couple this with the benefits of re-use, XML provides a significant ROI for localization of content.

### XML Supports Collaboration

Because XML is an open standard, it is a perfect medium for collaboration. An XML editor like Adobe FrameMaker can provide full-blown publishing capabilities, or an editor like FrameMaker XML Author® can provide just collaboration features at a lower cost. You can even find XML editors that work on the web, or you can use a text editor running in a programmer's integrated development environment. The important thing is that with XML you are free to mix and match your authoring tools, and you can choose the tool that is most appropriate for each individual contributor.

### Why Use Structured FrameMaker?

Structured FrameMaker is a full-featured XML publishing package. Not only does it offer tools to edit XML through a rich user interface, but it maps your XML to the powerful FrameMaker formatting capabilities you already know—you can use tools you already know to design the look of your final documents.

Both the structured and unstructured versions of FrameMaker excel in creating, editing, and producing complex content, so authors can easily create PDF files with linked bookmarks, cross-references, tables of contents, and indexes. Moving to structured FrameMaker gives you the best of both worlds—you can take full advantage of XML without giving up the powerful authoring environment you appreciate in unstructured FrameMaker.

### Authoring Visually

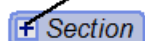
Structured FrameMaker gives you several different ways to look at your document. The document window is almost identical to the document window in unstructured FrameMaker. The document displays in WYSIWYG, showing the formatting you have designed for printed output.

Inside the document window, you have the option to display element tags for a structured document. The tag boundaries include controls to expand and collapse element content. For example, you can collapse entire sections to keep your focus on one specific section.





Showing element boundaries, with a collapsed section tag

In addition, FrameMaker includes the Structure View, which shows the document elements in a hierarchical

This section has been collapsed. Click the PLUS to expand it and show the content.



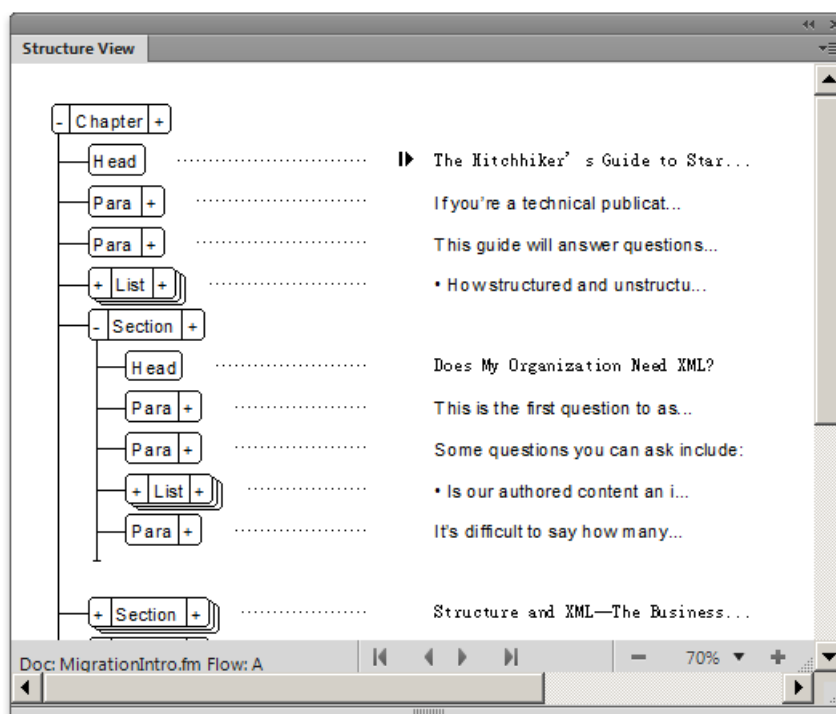
Section Head Expanding and Collapsing  
Heads / Head

 **Para** You can expand and collapse entire tags and their contents. This helps you focus on one part of the document.  **/ Para**  **/ Section**  **/ Chapter**

tree. You can expand and collapse the elements, and drag them to rearrange the document structure.

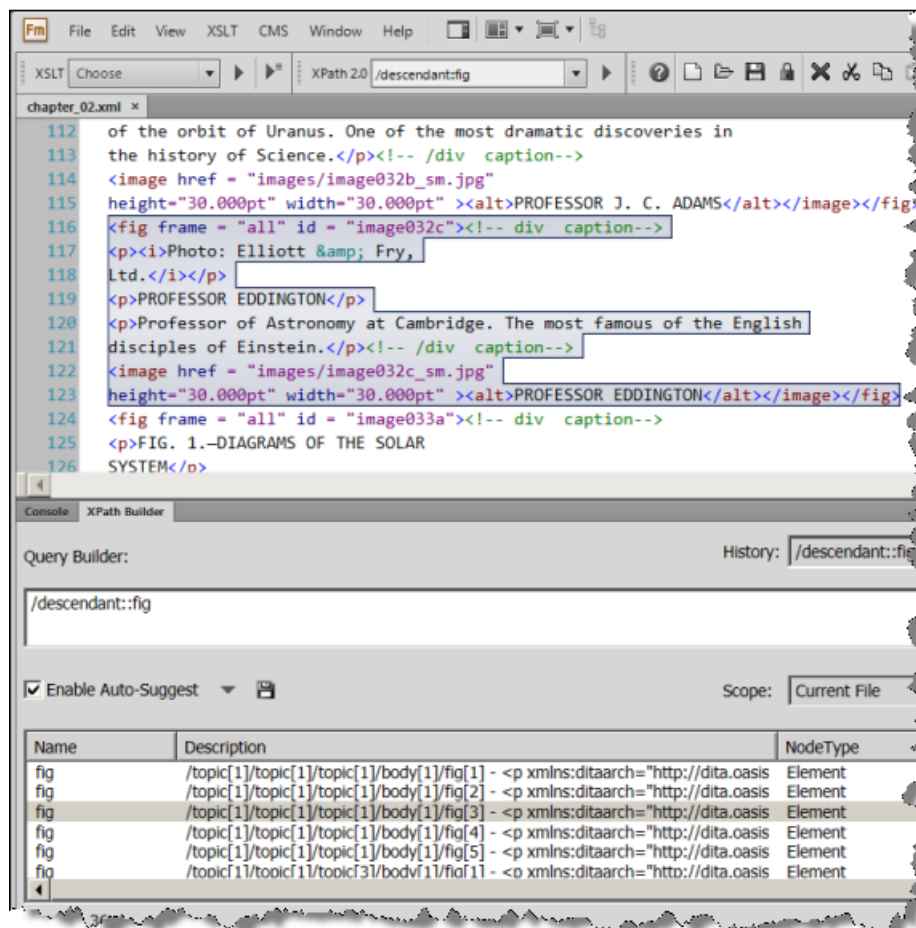
## The Structure View

## Editing XML Code



There are times when it's more useful to edit raw XML code. You can switch your document to the XML view to edit XML code, run XSL transforms on the document, or use XPath to search for specific types of content in the document. XML View is a powerful way to dig in and hack at the XML code of your document. Structured FrameMaker doesn't require you to work this way, but it doesn't stop you, either.

Editing in XML View, using XPath to find all figure elements in the document



### Excellent Print and PDF Output

Structured FrameMaker provides everything you need to create high-quality print and PDF output. Many other XML authoring tools do not include a high-quality print solution, and designing XML layout for print and PDF is much more complex than publishing to the web. Template design for structured FrameMaker is similar to what you already know for unstructured FrameMaker. With Structured FrameMaker you can apply your current knowledge and abilities to formatting XML documents for print or PDF output.

Saving XML as PDF is easier with structured FrameMaker than it is with any other tool. FrameMaker automatically creates bookmarks and links in PDF based on the elements in your document.

You can also use FrameMaker to generate a CSS style sheet that matches the document's WYSIWYG formatting. Any XML processor that supports CSS can use the style sheet to reproduce the look and feel of your document.

## Automatic and Enforced Formatting

In structured FrameMaker, authors don't format their documents with paragraph and character tags. Instead, they insert elements and element content into their documents. Document formatting is applied automatically based on formatting information that is embedded in the structured template. The structured template includes structure definitions in an Element Definition Document (EDD). This EDD references formatting information such as paragraph or character formats, and applies that formatting to elements in the document.

Element formatting is context-sensitive; that is, a single element can appear with different formatting in different structured positions. A single ListItem element, for example, can replace half a dozen (or more) paragraph tags. The following figure shows how formatting of the ListItem element changes based on its position and the value of the parent List element's Type attribute.

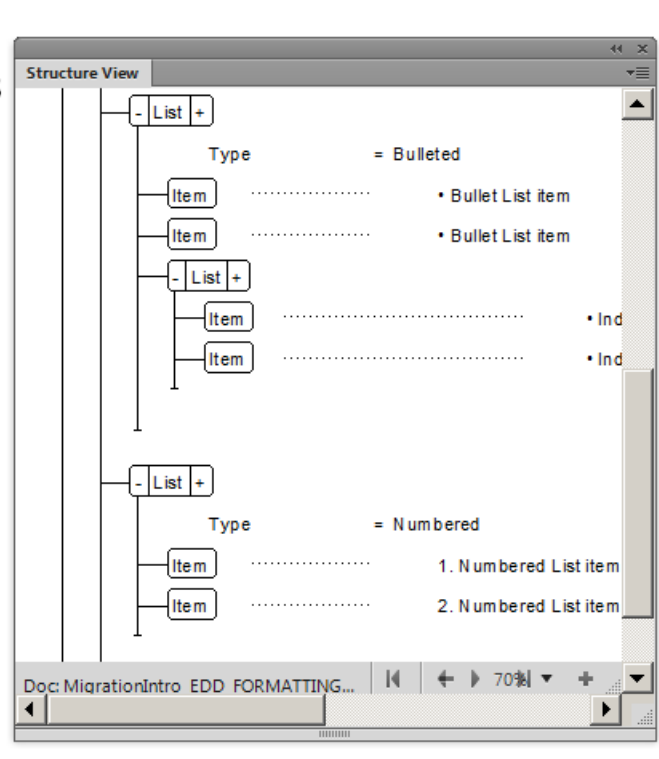
NOTE: Formatting in this example is controlled both by the nesting level of the ListItem and by the value of the Type attribute on the container element List.

List item formatting in different contexts

### Different List Types

Attributes set list type...

- Bullet List item
- Bullet List item
  - Indented item
  - Indented item
- 1. Numbered List item
- 2. Numbered List item



As you position information in the structure, the context-sensitive formatting updates immediately according to the element position. This dynamic formatting is extremely powerful when combined with FrameMaker's WYSIWYG interface.

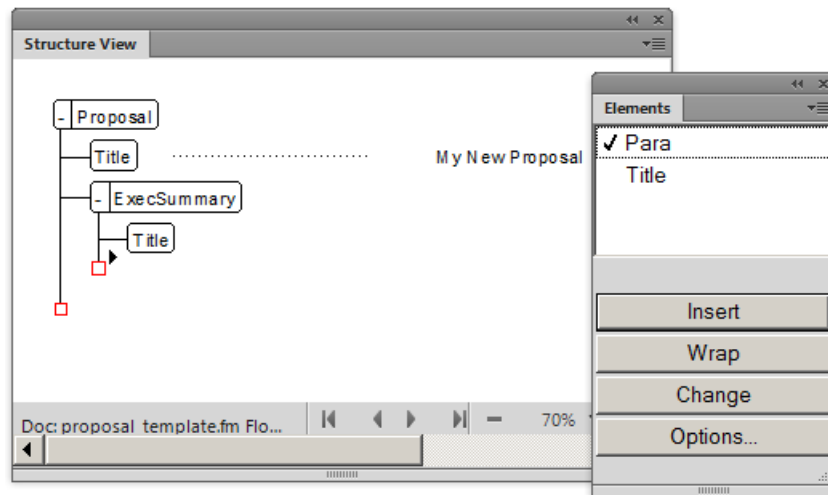
## On-The-Fly Validation (Guided Editing)

We have already seen that XML can include rules that determine a required order and hierarchy of elements. For example, a proposal document could require a Proposal element for the document root, followed by a Title element, then an ExecutiveSummary, a ProjectDescription, a Cost element and a Schedule. The ExecutiveSummary could require a Title, followed by any number of Para elements.

A document that omits one of these elements, or uses the elements in a different order, violates these rules. The document is said to be invalid. FrameMaker's Structure View shows in real time where required content is missing, and which elements are valid to insert at the current location.



FrameMaker's Structure View indicates that a required element is missing with a small red square. The Elements catalog indicates that a Para element is allowed here.



At the same time, FrameMaker doesn't tie you to always creating valid content. As you edit, you can add content out of sequence. FrameMaker will show where the document structure is invalid, but it will not stop you from working just to require that the document is always valid.

### Relatively Low Licensing Cost

FrameMaker gives you everything you need to build a structured authoring environment. Structured Applications automatically recognize document types, and load XML into the appropriate document templates. These templates give you a WYSIWYG authoring environment that you can use with Structured FrameMaker, or deploy to contributors using FrameMaker XML Author. The template automatically applies formatting to your document according to the XML structure.

Structured FrameMaker includes tools to process the XML. These include integration with XSLT (stylesheets to transform your XML), integration with the DITA Open Toolkit, XPath processing, Structured FrameMaker read/write rules, FrameMaker ExtendScript, and the FDK.

You also get tools to create your Structured Application, and tools to convert unstructured documents to structure. If you currently use unstructured FrameMaker, then you already have all these capabilities at hand. Why not set the preference to use the Structured FrameMaker product interface, and take advantage of them?

### Why Use FrameMaker XML Author?

Structured FrameMaker gives you the tools to create a full XML authoring environment, as well as the authoring software and the capability to print or generate PDF versions of your documents. What if some members of your team only need to add content to an existing XML publication? For these team members, FrameMaker XML Author is the perfect solution.

For example, assume you need to include product specifications in your final publication. You might want product engineers to directly maintain the product specification documents, because they own that information. With XML Author, you can give them access to Structured FrameMaker templates, and have them work with an already defined FrameMaker structured application. They can then author the content in valid XML that you can include in the final publication.

XML Author supports enough features to create and edit XML documents that comply with a defined structured application. At the same time, it's fully compatible with the Structured FrameMaker templates you have created. And it supports any other processing you have set up in a structured application, including XSLT processing, read/write rules, or any FrameMaker plug-ins you may have included in your work flow.

Licensing cost for FrameMaker XML Author is even lower than the cost for FrameMaker. This makes XML Author an excellent tool to promote collaboration within your organization.

## Components of a Structured Authoring Environment

For unstructured FrameMaker, you set up the authoring environment by implementing your document templates. Once you define a template and distribute it, authors are ready to begin work. With structured FrameMaker the authoring environment is more powerful, and you must implement and deploy more files to put the environment together.

### Authoring Only in Structured FrameMaker

You can implement structure in FrameMaker without providing support for XML editing. If you have no immediate need for XML, you can still get value from structured authoring by imposing a structure on your publications, formatting documents automatically, and using attributes to filter your content.

To establish a basic structured authoring environment in FrameMaker, you must create an element definition document (EDD) that declares the elements in your structure. Additionally, you need to link the element definitions with formatting information. There are three ways to link elements in the EDD to formatting:

- *Formatting template*

You can assign formatting based on the various tags (paragraph tags, character tags, and so on) in a template. If you already have a formatting template, you can reuse the tags in that template.

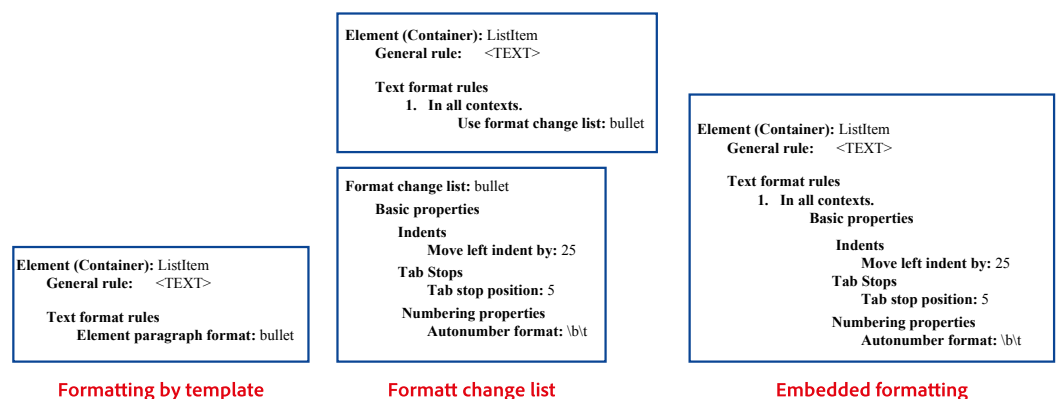
- *Format change lists*

You can create named formatting definitions in the EDD. These format change lists provide partial formatting specifications and inherit the rest from a few base paragraph tags. For example, you can specify one Heading paragraph tag that specifies font size and indent for H1. For nested heading elements, you can call a format change list that uses the same paragraph tag, but reduces the font size and increases the indent.

- *Embedded formatting*

You can specify formatting by writing the formatting into each element definition.

Comparing three ways to specify bullet formatting in the EDD



If you choose to use format change lists or embedded formatting, you will still need templates to set up basic formatting such as master page layouts and cross-reference formats.

Once you create the EDD and specify your formatting in a template, you then combine the two components to create a single, structured template. To do this, you open the formatting template and import element definitions from the EDD (using File>Import>Element Definitions). The resulting file is your structured template which you distribute to document authors. Remember to keep a separate copy of the EDD so that you can make updates and import them into your set of templates.

These steps set up structured authoring that doesn't necessarily use XML. There is value in using structure alone. The EDD enforces structure rules and automates formatting. Authors can reorganize the document by dragging entire sections to different locations in the document tree. If you use FrameMaker plug-ins or ExtendScript add-ons, these custom tools can operate on the document content more easily thanks to the structure information captured in it.

### **Structured Authoring and XML**

To support full XML editing with Structured FrameMaker, you provide the EDD and template, but you must also provide some other files.

- *XML structure definitions*

A document type definition (DTD) or Schema file provides structure rules for XML that match your FrameMaker element definitions. Unlike the EDD, DTDs and Schema files do not provide formatting information.

- *Read/write rules*

These rules control translation from FrameMaker structure to XML and back. Specifically, read/write rules help manage conversion of tables, cross-references, graphic attributes, and other complex formatting.

- *XSL transform templates*

Optionally, you can deliver a set of transforms that authors can use when writing or publishing documents.

- *Entity specifications*

Many XML standards use extra files to specify additional structure, or to specify handling of special characters. You need to specify locations for these files so the structures application can use them.

- *FrameMaker API client*

The FrameMaker Development Kit (FDK) customizes and extends FrameMaker processing. Many structured applications include a plug-in .dll file.

You deploy these files to your authoring environment, and specify their locations in a Structured Application file. FrameMaker reads this file on startup, and uses these files when you edit and publish your XML documents.

If this seems complicated, don't panic. In the first place, FrameMaker comes with a default Structured Application file, and it specifies structured applications to work with a number of XML standards, including DITA, DocBook, and XHTML. You can use these applications out of the box, or customize them to change the look of your publications. To make these customizations you will probably only need to change the formatting of structured application templates.



NOTE: There are a number of open source XML databases that you could use to extract specific content from a collection of XML files. For example, you could use a database to extract cost estimates from all the saved proposals.

Before building the structured application for your proposals, you should analyze existing proposals to identify their components and structure.

## Reviewing existing proposals



Based on analysis, you can create a content map. Analysis of your proposals results in the following sequence:

- Title
- Executive Summary
- Executive Summary:Title
- Executive Summary:Paragraph (one, only)
- Project Description
- Project Description:Title
- Project Description:Paragraph (one or more)
- Cost
- Cost:Title
- Cost:Paragraph (one or more)
- Schedule
- Schedule:Title
- Schedule:Paragraph (one or more)

This is a simple example—you could probably build the structured application for this proposal without the benefit of a formal analysis. For larger projects you will see that formal analysis is a crucial first step.

### Choosing an EDD Strategy

The EDD defines the structure that FrameMaker will use when it loads the structured document. It also maps the structure to FrameMaker formatting. There are several ways to start building your EDD. You can:

- Create the EDD yourself, starting from scratch
- Import a DTD or Schema to create an EDD with structure definitions—you then add formatting rules to those structure definitions
- Modify an existing EDD—FrameMaker ships with a number of EDDs you can use as starting points
- Use an existing Structure Application—FrameMaker ships with Structure Applications for standards such as DITA, XHTML, S1000D, and others. You can use those applications as-is, or modify the templates and EDDs to refine the document formatting.

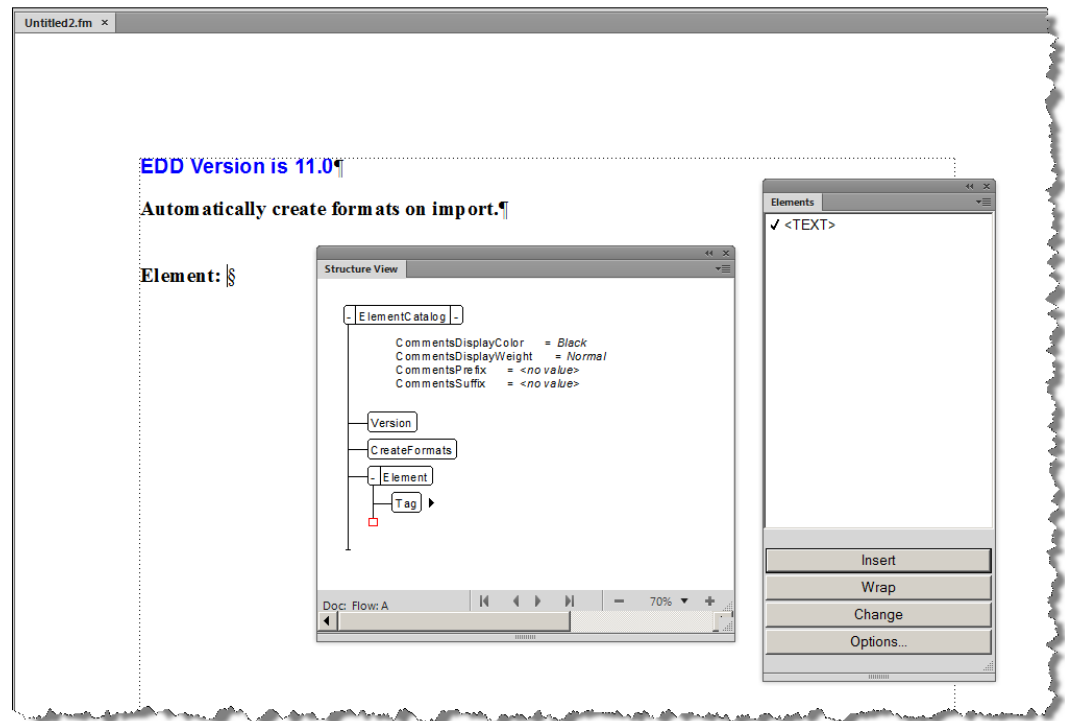
For the Proposal project, we will create an EDD from scratch. This best illustrates the workings of an EDD within a Structure Application.

### Building the Proposal EDD

Now that we have analyzed our proposals and decided on the way forward, let's build the EDD. To do this:

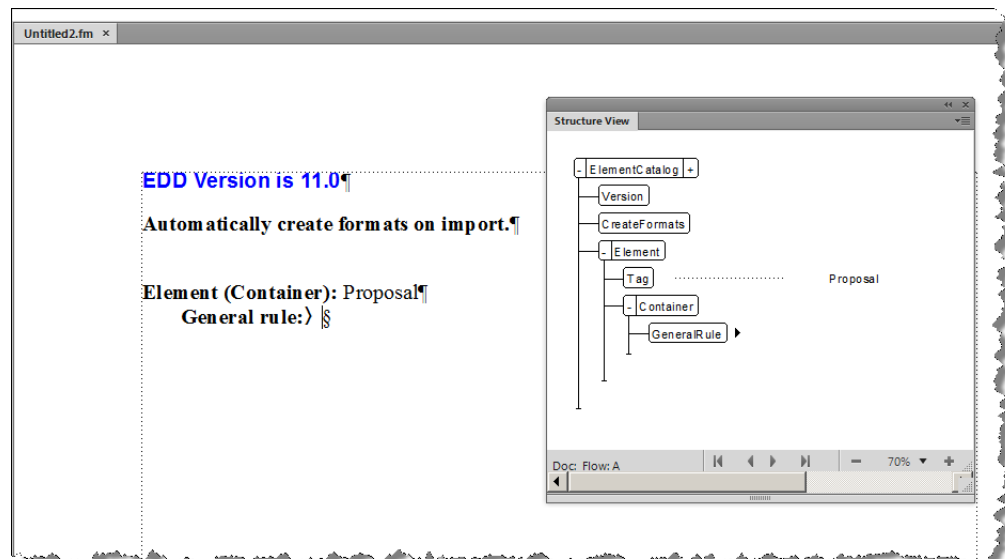
1. Make sure you're using Structured FrameMaker. If the menu bar doesn't include the Element and StructureTools Menus, you must switch to the Structured FrameMaker product interface. Don't worry, you can always switch back to unstructured, and this will not affect your ability to edit unstructured documents. Choose *Edit>Preferences>Global>General*. For Product Interface, choose *Structured FrameMaker*.
2. Create a new EDD  
Choose *StructureTools>New EDD* to open a new EDD file.
3. Open the Structure View and Element Catalog. If they are not already open, choose *StructureTools>Structure View*, and then choose *Element>Element Catalog* to open them.





4. In the Structure View, click to the right of the red box (which indicates that additional information is required). Notice that the contents of the Elements catalog change because of the new cursor location. In the Elements catalog, select Container and click Insert. The Container element and a child GeneralRule element are inserted (as shown in the following figure).

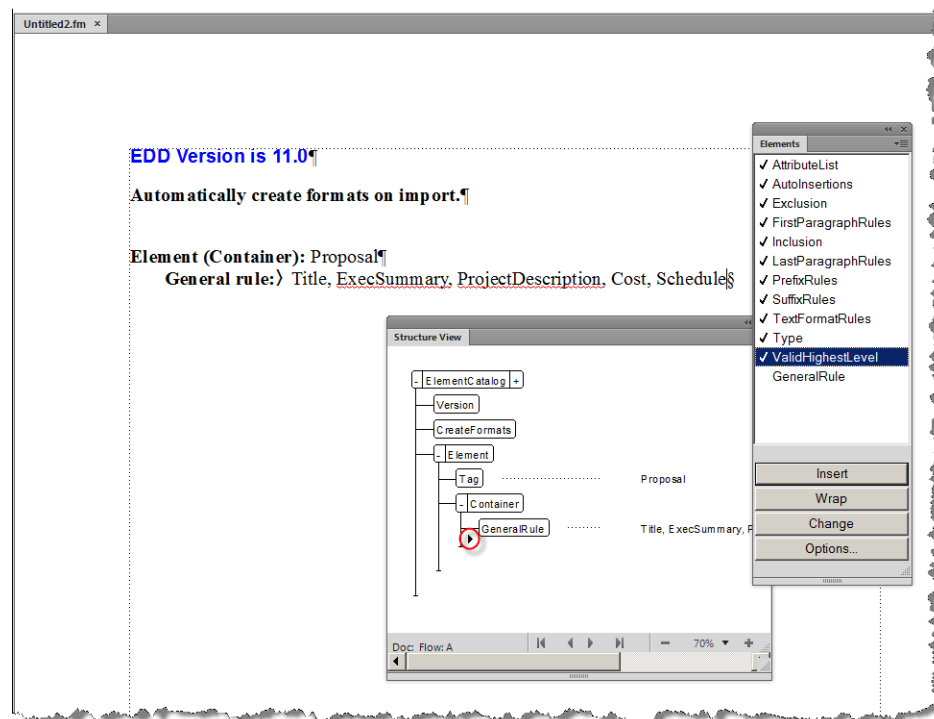
#### Adding the general rule for Proposal



5. Click to the right of Tag and type "Proposal" to give this element a name. Note that element names cannot contain spaces.
6. Enter the Proposal element's general rule to define the proposal content. This should be the top-level elements we identified in the document analysis. Click to the right of GeneralRule and enter:  
Title, ExecSummary, ProjectDescription, Cost, Schedule

7. Insert a ValidHighestLevel element as a sibling of the GeneralRule element. To do so, click underneath the GeneralRule element to position your cursor as shown in the following figure, select the ValidHighestLevel element in the Elements catalog, and then click Insert.

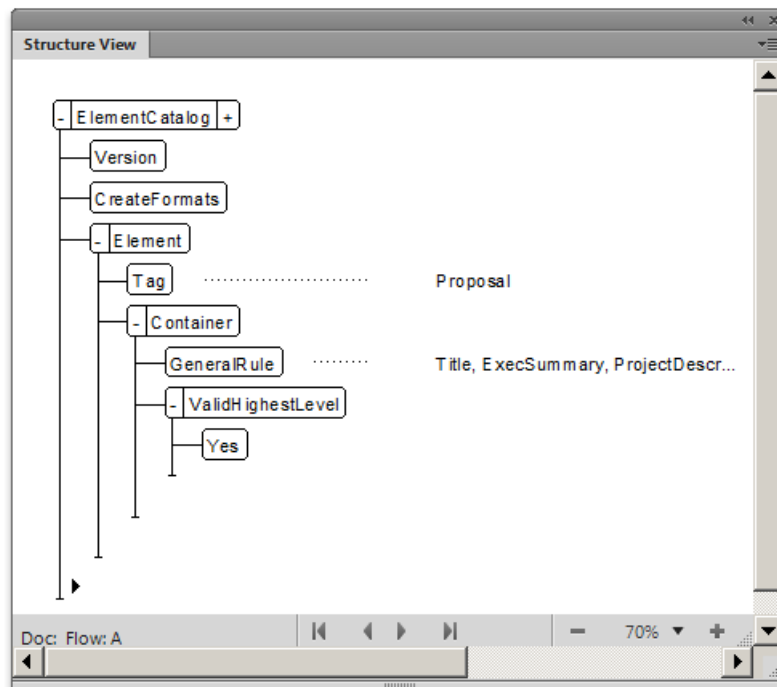
#### Inserting the ValidHighestLevel element



After setting it as the highest-level element, the definition for the Proposal element is complete. Now you must define each of the child elements: Title, ExecSummary, ProjectDescription, Cost, and Schedule.

To define the first child element:

1. Position the cursor at the bottom of the structure as shown:



- Using the Elements catalog, insert an Element bubble. Name the element ExecSummary, make it a container, and specify the following as the general rule: Title, Para+

Defining the ExecutiveSummary element

**EDD Version is 11.0**

**Automatically create formats on import.**

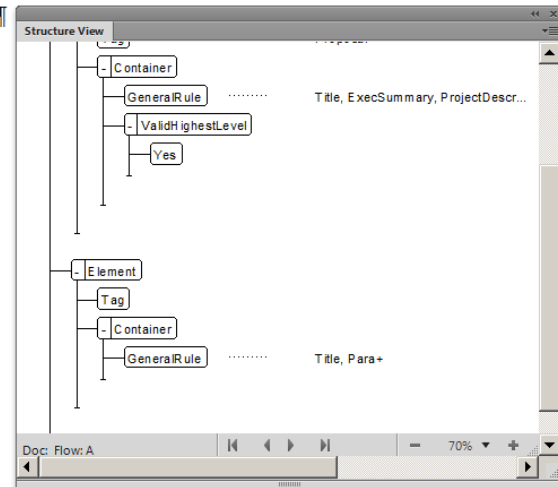
**Element (Container): Proposal**

**General rule: Title, ExecSummary, ProjectDescription, Cost, Schedule**

**Valid as the highest-level element.**

**Element (Container):**

**General rule: Title, Para+**



Repeat these steps to define the remaining elements. Give them the following names and general rules:

Element	General Rule
ProjectDescription	Title, Para+
Cost	Title, Para+
Schedule	Title, Para+
Title	<TEXT> (Type the word TEXT, surrounded by angle brackets)
Para	<TEXT> (Type the word TEXT, surrounded by angle brackets)

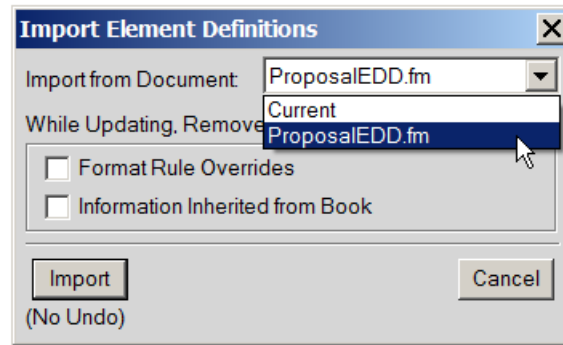
Be sure to save your EDD as ProposalEDD.fm.

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

### Testing the EDD

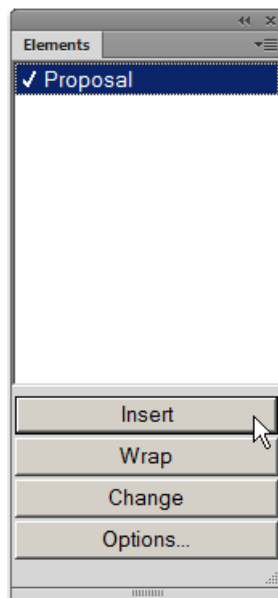
To test the EDD, import it into a document and then verify that you can create the structure you expect. At this point the document will not have any formatting, but you can verify that the structure is working as expected.

- Create a new, blank document. Choose *File>New>Document*, and then click Portrait.
- Import the EDD into the new document. Make sure the new document and the EDD are both open. With the new document active, choose *File>Import>Element Definitions*. In the Import From drop-down, select your proposal EDD, then click Import. If your EDD doesn't appear in the drop-down list, make sure that the EDD file is open, and that you have saved it.



3. Now that the element definitions are imported in your document, you can start adding structure to it. Click in the new document and open the Element Catalog. It will show the Proposal element as the only valid element at that location. Insert the Proposal element, and then insert a Title element.

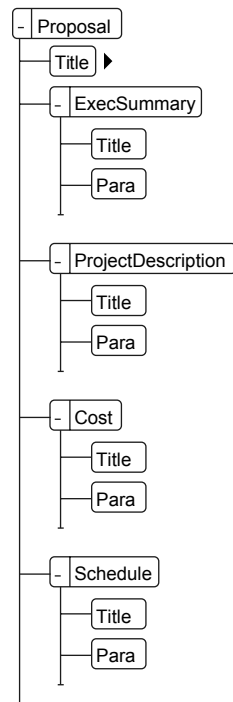
The Element Catalog, showing the Proposal as the valid top element in the new document



4. Continue inserting elements until the proposal structure is complete. You can add text content if you like, but you don't need to at this time. The final result will appear as follows in the Structure View:

## The complete Proposal structure

5. Save your work! Save the new document as structured\_proposal.fm



If your structure doesn't match this figure, go back to your EDD and correct it. Then import the EDD into your document again.

### Adding Formatting to the EDD

You have now built an EDD that provides structure for a simple proposal. However, when you type content, no formatting is applied. By default, FrameMaker applies a basic Body paragraph format to any text content. This section describes how to provide formatting, and how to automatically insert the correct text for the various titles.

Before assigning formatting to the EDD, you should define the basic formats you will use in your document. Type placeholder text in your proposal document and use standard FrameMaker techniques to modify the Body and Heading1 paragraph formats. Make sure to Apply All to get your changes into the paragraph format catalog.

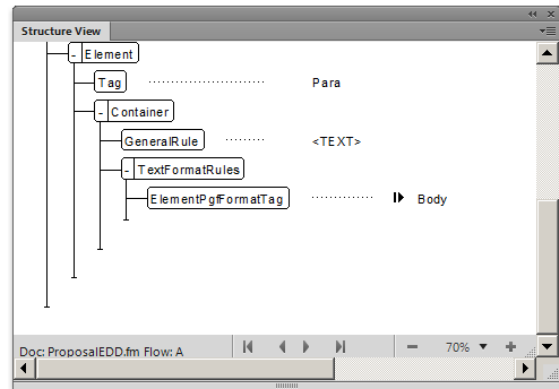
Choose *Format>Paragraphs>Designer* to make your changes. The EDD will use Body and Heading1 as the basis for all other formatting. When you have your basic Body and Heading1 formats the way you want them, it's time to set formatting in the EDD.

### Specify Formatting for Para

In proposalEDD.fm, modify the Para element definition to include a formatting rule. To specify that Para should always use the Body paragraph tag, click under the GeneralRule element, add a TextFormatRules element, and then add an ElementPgFormatTag element. Type Body as the text for the ElementPgFormatTag element.



**Element (Container):** Para¶  
**General rule:** <TEXT>¶  
**Text format rules**¶  
**Element paragraph format:** Body§



### Specify Formatting for Title

For this element you need more complex formatting rules. Title should automatically display section titles, such as Executive Summary, Project Description, and so on. You must write a context rule that specifies what text to display for each type of heading, and specify that Title uses the Heading1 paragraph tag.

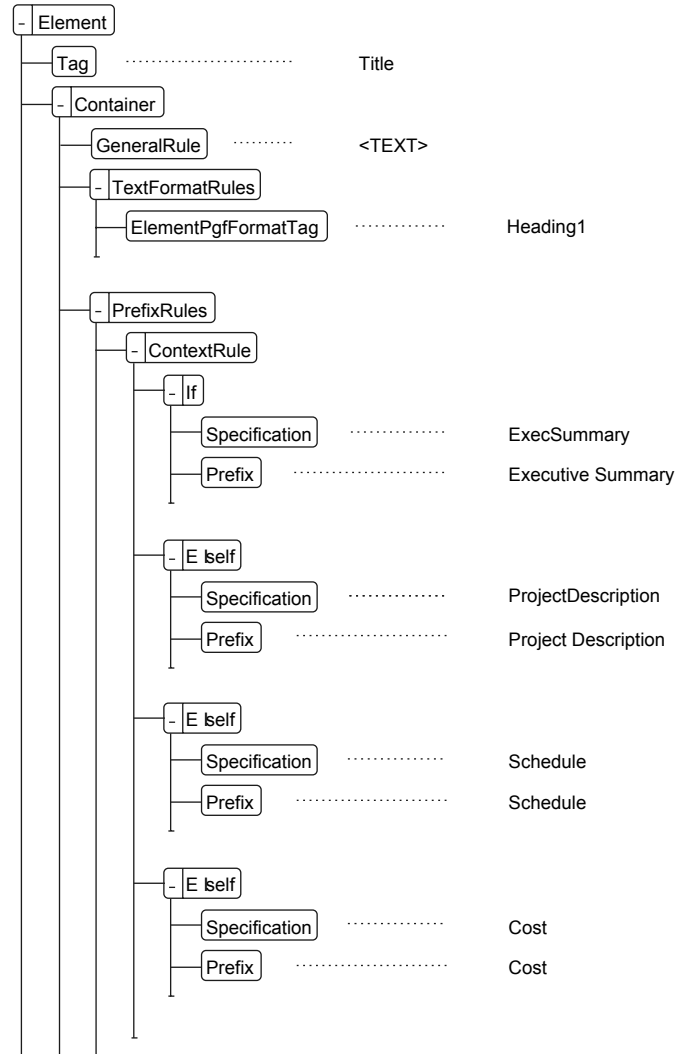
In proposalEDD.fm, modify the Title element definition to use the Heading1 paragraph tag. Add the same TextFormatRules and ElementPgFormatTag elements as you did for the Para element, but give the format name, "Heading1". Remember that in FrameMaker paragraph format names are case-sensitive. For example, "Heading1" and "heading1" are not equivalent.

Next, add a prefix rule to the Title element. Prefix rules specify text that should appear at the beginning of the element. Based on the Title's position, you'll specify which text to display. This will automatically provide the titles according to the proposal structure.

To add a prefix rule to the Title definition in the EDD:

1. Position the cursor in the Title element definition to insert a bubble within Container, and after TextFormatRules. This will be a child of Container, and a sibling of TextFormatRules.
2. Insert a PrefixRules element.
3. Insert a ContextRule element. The If and Specification elements are inserted automatically. The context rule will determine which title content you want.
4. For Specification, enter ExecSummary—This rule assigns text to the Exec-Summary title.
5. Position the cursor as a child of If, after Specification, and insert a Prefix element.
6. For the Prefix element, type Executive Summary
7. Add Elself elements to the context rule to repeat these steps for the other elements that need titles. Create Elself statements for ProjectDescription, Schedule, and Cost. Give each prefix the appropriate title text.

Your results should be similar to the following:

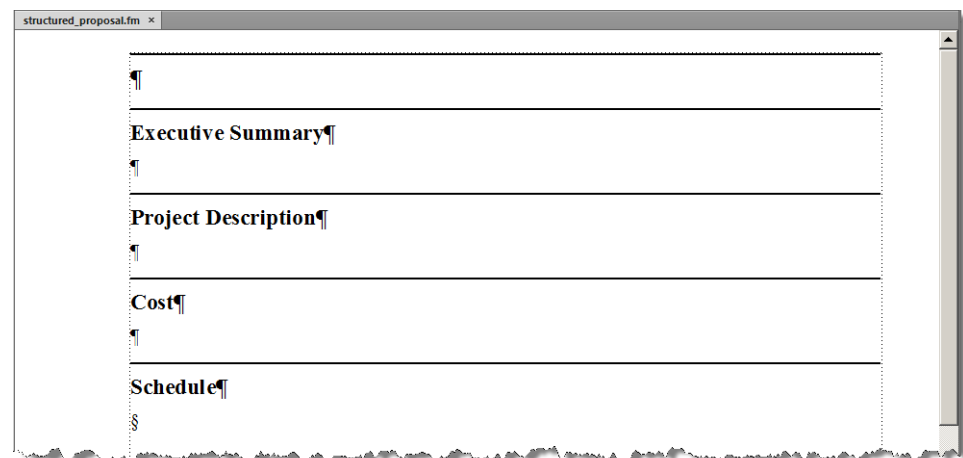


Title formatting rules

Be sure to save your work!

### Test The Formatting

To test your formatting, import the changed EDD into your existing proposal document. Each section should now display the title text you specified. But the main proposal title should be blank (you can type whatever title you want).



## Building a Structured Application

So far you are set up to create a proposal as a structured document. But to set up a work flow that includes authoring and storing the proposals as XML, you need to build a structured application. This application lists the components of the structured authoring environment—one of those components will be a template document that uses the EDD you just created. But you will also need other components.

For this exercise we'll build the minimal structured application, which includes the following files:

- DTD, to declare structure rules for the XML
- Template file, to combine structure rules and formatting in FrameMaker

When we have the DTD and template file in place, we'll add them to a structured application, and read that application into your FrameMaker session. Then we'll test it for editing XML documents.

### Creating a DTD

We can create an XML DTD from the EDD you just created. To do this:

- Open the EDD file you just created (ProposalEDD.fm).
- With the EDD file active, choose *StructureTools>Save As DTD*. Specify the file name (proposal.dtd) and where you want to save the file, and then click Save.
- In the Use Structured Application dialog box, leave the default selection, <No Application>, and click Continue.
- In the Select Type dialog box, select XML and then click OK. The DTD file is written out to the location you specified.

That's all there is to it. Remember the DTD file location so you can add it to the structured application later.

### Creating a Structured Template

The application will use your structured template when you open an XML file. FrameMaker reads the XML structure, then uses the template's EDD to map the structure to the template's formatting.

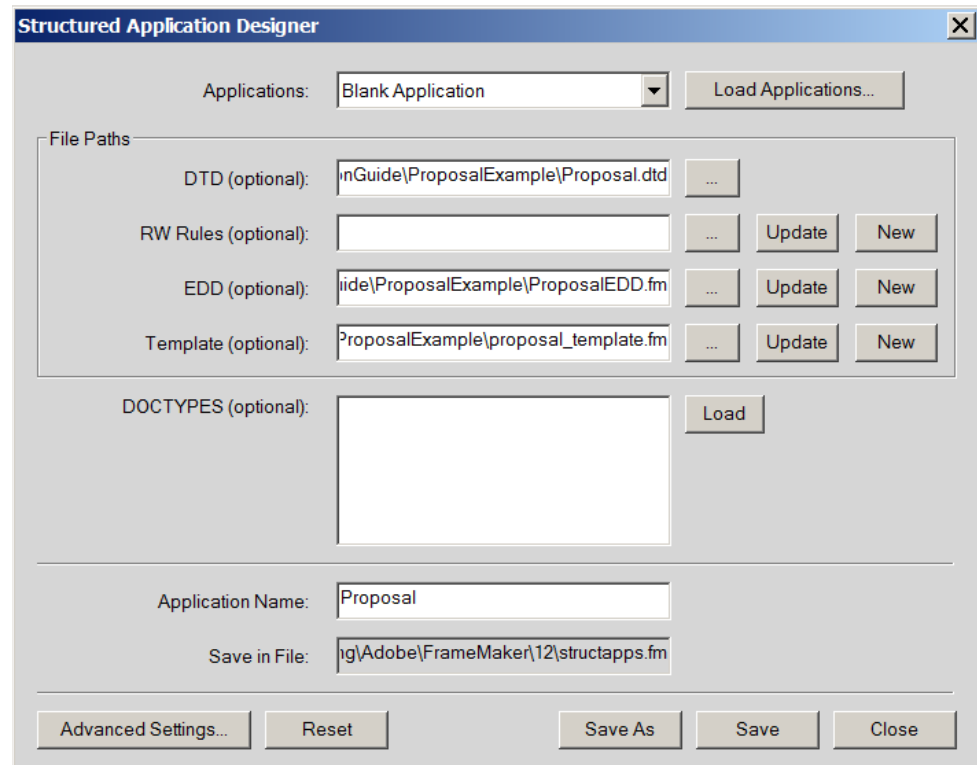
To create a structured template:

- Open the structured proposal document you just created (structured\_proposal.fm).
- Delete all the content from the template. This template will now be empty.
- Use *File > Save As* to save a version of the file named proposal\_template.fm.

Now you have a structured proposal template, ready to use. Remember that this template already has the EDD information imported into it. If you decide to change the EDD, you can always import those changes into the template file at a later date.

### Creating a "Proposal" Structured Application

Now that you have the DTD and template file, you can make a structured application. To do this, you will specify the DTD and template files in the Structured Application Designer to create the application entry in the Structured Applications file. When that's complete, you will load the modified application file into your current session. For subsequent sessions, FrameMaker will load the new application automatically.



- Choose *StructureTools>Structure Application Designer*.
- Specify the locations of the DTD and the template files. Then give a name for the structured application. In the above figure, the name is "Proposal".
- Click Save. By default the designer saves your application in the roaming structured applications file. After saving your application, click Close.

### Testing the XML Authoring Environment

Now that we have defined the structured application for a proposal, we can edit the proposal as an XML file. We can use FrameMaker to save the structured document as XML, and from then on we can keep the data in XML. We can open it in FrameMaker whenever we like, or edit the file in any other XML editor.

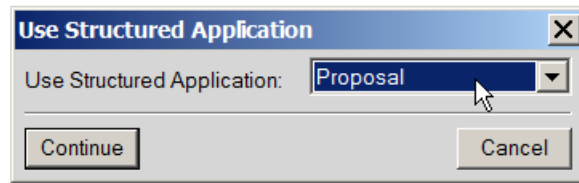
The first step is to load the Proposal structure application into the FrameMaker session. When you saved the application definition, by default FrameMaker saved it to the Application Definitions file for your user account. Whenever you log into FrameMaker, that application definitions file loads, and you can use the Proposal application.

To load it for the first time, we'll load the application definition without restarting FrameMaker:

1. Choose *StructureTools>Edit Application Definitions* to open the Application Definitions file. This file is named structapps.fm. At the bottom of the file, in the last section before "Defaults", you should see your Proposal application definition.
2. With the Application Definitions file active, choose *StructureTools>Read Application Definitions*. This loads the new definition into your FrameMaker session. Now you can close the structapps.fm file.

The next step is to open your structured proposal and save it as XML. With the structured proposal active, choose *File>Save As*. At the bottom of the dialog box, set the Save As Type to XML, then click Save. A dialog box appears asking you which structured application you want to use. Be sure to choose the Proposal application. (You will see that FrameMaker ships with a long list of ready-made structure applications.)

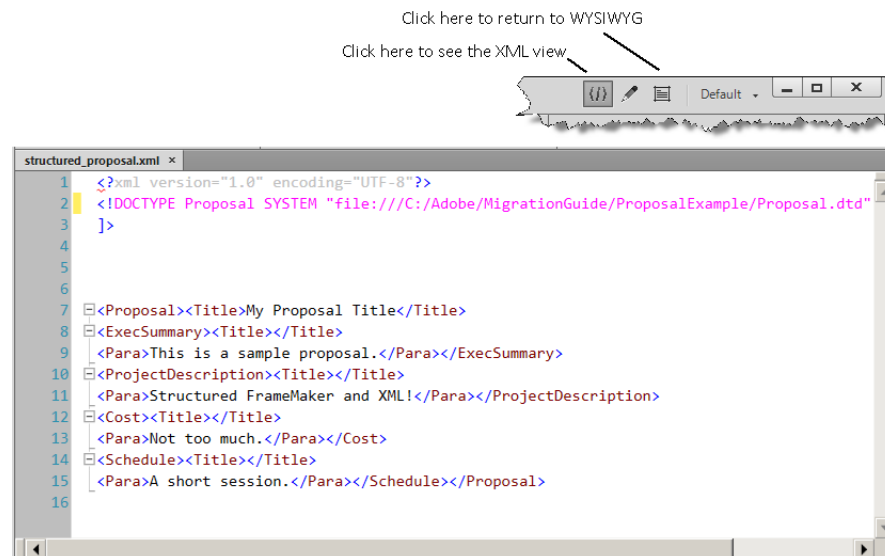
## Choosing the Proposal application



FrameMaker saves an XML version of the file to your disk. Go ahead and close the structured\_proposal.fm file (the FrameMaker document) and open structured\_proposal.xml. The same dialog box appears asking which structured application to use—be sure to use Proposal.

The XML version of the document should look just like the structured .fm document. But this document is in XML. Add some content to the document. Then with the XML document active, switch to XML view to see the raw XML for your proposal document. You can also open the XML file in a text editor if you wish. Use the XML view to run transforms on the content, or use XPath to search through your document.

## Viewing the proposal XML



## Wrap-up

You've just walked through the process to design, implement, and use a structured application to edit XML files. This was a very simple case, but it illustrates the basic steps. You should be able to see that Structured FrameMaker is a powerful tool for editing XML documents, and producing formatted output.

There is some effort in creating a structured application, but once you have the application in place, the benefits to your work flow become apparent. If you want to work with XML, Structured FrameMaker gives you everything you need—unparalleled formatting capabilities, best of breed document processing, and a full range of XML processes including XSLT, XPath search, and integration with the DITA Open Toolkit.

To recap, the steps to set up XML editing in FrameMaker include:

- Perform content analysis
- Decide on structure
- Implement the XML structure (a DTD or Schema)
- Implement the Structured FrameMaker template, including structure (the EDD) and formatting
- Create the structured application
- Test the application's handling of XML and formatting



When planning a structured application, you should consider a few points.

Document structure can be strict or loose. Very strict structure rules may disallow element combinations that authors will need. These limits can make it difficult to author a document. On the other hand, a very loose structure is necessarily more complex, and the DTD and EDD might be difficult to maintain. This is because more elements are required as you increase the number of possible combinations and structures. Finding the balance between these extremes can be difficult. This is one reason that many organizations prefer to use an existing standard.

There are many existing XML standards you can use. One significant advantage of using standards is sharing—you can easily share your files with any other organization that uses the same standard. Another advantage is that much of the work to implement your editing environment is already done for you. Structured FrameMaker ships with structured applications already in place for many standards, including DITA 1.1 and 1.2, Docbook, S1000D, and XHTML. Before investing in creating your own DTD, it's always a good idea to look at these standards and see whether they will suit your purpose.

### **Migrating Unstructured Files to Structure**

Migrating legacy documentation to XML is a common requirement once you decide to work in XML. If structure and XML can add value to your planned information assets, then it follows that migrating legacy assets to XML can add even more overall value.

Structured FrameMaker includes a conversion table utility that you can use to transfer unstructured FrameMaker files over to whatever structure is defined in your structured application. Once converted to structure, you can save the documents as XML to complete the migration.

The conversion table maps formatting components in your document to an element structure. The components it can map include paragraph tags, character tags, markers, cross-references, and table components. You can create a conversion table that works with a given template, and automatically generate structured copies of any documents that use that template.

The conversion results you can achieve depend on the following factors:

- consistency—Documents that use a FrameMaker template consistently, with few or no formatting overrides, will convert better than documents that are full of overrides and custom paragraph or character tags.
- Similarity between unstructured and structured documents—When you perform your document analysis and set up the EDD for your structured application, you can choose to implement a structure that is similar to your legacy documents. The more similarity you can achieve, the easier the conversion process.

**TIP:** What if your legacy documents don't match your desired structure? For example, what if you want to use an XML standard that is unlike your legacy documents? In that case, you can create an interim EDD that matches your document structure, and then convert the documents to that XML. Once in XML, you can use existing tools such as XSLT to further modify your XML documents and bring them into compliance with the standard you chose.

The steps you take to migrate your legacy documents include:

- Clean up the legacy documents
- Create the conversion table
- Generate your structured documents
- Save the structured documents as XML

The following sections illustrate these steps.

## Cleaning Up Legacy Documents

Unstructured FrameMaker authors are not strangers to the idea of structured documentation. FrameMaker includes report and book templates that encourage you to order documents via headings, lists, and character tags. For example, headings create a hierarchical structure which is not unlike the structure you would declare in XML.

The first step for conversion is to make sure your documents use the templates as they were designed. You should check the following:

- Make sure variable definitions are up to date. In addition, it's best to use variable names with no spaces in them.
- Update all references in your documents, including cross-references and text insets. Correct any unresolved references.
- For documents with conditional text, show all conditions. Structured FrameMaker creates XML that maintains your conditional text settings. But when converting to structure, you need to show all conditional text to make sure it gets wrapped correctly in structure elements.
- Make sure the ordering of your content is correct. For example, heading structure should not skip heading levels. Heading1, Heading2, Heading3 is correct, but Heading1, Heading3 is not.
- Eliminate format overrides for paragraph, character, and table formats. You can use the Find/Change dialog box to search for these overrides.

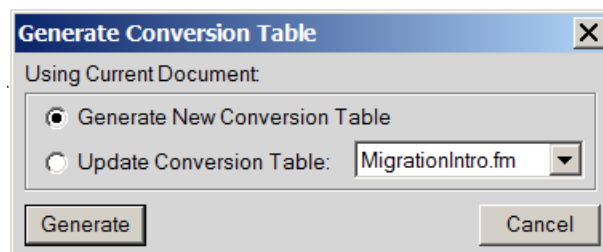
## Creating the Conversion Table

The conversion process creates structured elements from FrameMaker formatting components such as paragraph tags, character tags, markers, cross-references, and table components. To begin the conversion, select a document that is representative of your typical content. This document should contain examples of all of the formatting tags that would occur in your documents. These tags should be used in their logical sequences (as they would occur in documents), so a formatting template that shows examples of each paragraph tag in alphabetical order is not a good example document.

To begin building the conversion table:

1. Open the example document you want to use, and save a copy of it. This will be your working document.
2. Open the EDD for the structured application that you are converting to, and import the element definitions into your working document.
3. With the working document active, choose *StructureTools>Generate Conversion Table*. Select *Generate New Conversion Table*, and then click *Generate*.

Creating a new conversion table



FrameMaker scans the working document and creates a list of the formatting components that occur in this document. Any tags that are defined in a format catalog but not used in the document are not included in this list.

Wrap this object or objects	In this element	With this qualifier
P:ReportTitle	ReportTitle	
P:Heading 1	Heading 1	
P:Body	Body	
P:Heading 2	Heading 2	
P:Bulleted	Bulleted	
P:BulletedCon	BulletedCon	
P:Heading 3	Heading 3	
P:TableTitle	TableTitle	
P:CellHeading	CellHeading	
P:CellBody	CellBody	
P:Extract	Extract	
P:Figure	Figure	
P:Equation	Equation	
SV:Table Continuation	Table Continuation	
Q:Medium	EQUATION	
G:	GRAPHIC	
TT:	TITLE	
TH:	HEADING	
TB:	BODY	
TR:	ROW	
TC:	CELL	

Paragraphs formats  
converted to  
elements

Variable  
Equation  
Graphic

Table components

The entries in the conversion table identify the type of formatting object to map, and give default names for the elements. In the above example, a paragraph tagged as ReportTitle will get wrapped in the ReportTitle element, and a variable named Table Continuation will get wrapped in the TableContinuation element.

If you use the default generated table, the resulting structure will probably not match the EDD for your structured application. To change the table, you will add structure rules in the first column, and change the names of elements in the second column. You can also add qualifiers to fine-tune the conversions.

For example, you can change the P:Body entry to wrap body paragraphs in a Para element. You can also change the table to wrap P:Heading1 in a Heading element, and then wrap any Heading element and the following Para elements in a Section, as follows:

P:Heading 1	Heading 1	
P:Body	Para	
P:Heading 2	Section	
P:Heading 2	Heading 2	

As you make these changes, you should save and test the table. To test it, make your working document active, then choose *StructureTools>Utilities>Structure Current Document*. Select the conversion table document in the drop-down list, and then click Add Structure. FrameMaker creates a new, untitled, structured document. Display the structure view for that document and verify that you got the structure you want.

Keep on refining and testing the conversion until you are satisfied with the resulting structure.

If you need to map several paragraph tags to the same element and then wrap them into different parents, you use the third column for a qualifier. It's common, for example, to have a `ListItem` element that's used for both bulleted lists and numbered lists. Once the bullet and step paragraphs are wrapped in the `ListItem` element, you need a way to distinguish whether they belong in `OrderedList` or `UnorderedList`. To make this distinction, you use the qualifier column, as shown in the following example:

Wrap this object or objects	In this element	With this qualifier
bullet	<code>ListItem</code>	<code>ul</code>
step1	<code>ListItem</code>	<code>ol</code>
step2+	<code>ListItem</code>	<code>ol</code>
<code>E-ListItem[ul]+</code>	<code>UnorderedList</code>	
<code>E-ListItem[ol]+</code>	<code>OrderedList</code>	

There are many other details about creating conversion tables. FrameMaker ships with the online Structure Application Developer's Guide, which includes all you need to know about creating conversion tables and wrapping elements according to content rules.

### Conversion Example

The best way to illustrate the way a conversion table wraps unstructured content in elements, and builds the document into a hierarchical structure, is to show an example. We can start with an unstructured outline, taken from the standard templates, generate a conversion table, then modify it to build a tree of elements that correctly represents the outline's hierarchy.

#### Open the Outline Document

First, let's start with the outline document. Open the template browser (choose *File>New>Document*, then click *Explore Standard Templates*), and open a sample of the Harvard outline.

The sample Harvard outline

## World of Text Composition

Paul J. Murphy

### I. Introduction

**A. Mirum est ut animus agitatione motuque corporis excitetut.**

1. Iam undique silvae et solitudo ipsumque illud silentium.  
Mirum est ut animus agitatione motuque corporis excitetut. Non est quod contemnas hoc studendi genus. Ego ille quem nosti montibus quam apros et quidem pulcherrimos cepi.
2. Quod venationi datur magna cogitationis incitamenta sunt.
  - a) Erat in proximo non venabulum aut lancea.  
Ego ille quem nosti apros et quidem pulcherrimos cepi. Mirum est ut animus agitatione motuque corporis excitetut.
  - b) Experienis non Dianam magis montibus quam Minervam inerare.

**B. Sed stilus et pugilares meditabar aliquid enotabamque.**

1. Si manus vacuas, plenas tamen ceras reportarem.
2. Ridebis, et licet rideas.
  - a) Meditabar aliquid enotabamque non tamen ut omnino ab inertia mea et quete discederem.
  - b) Mirum est ut animus agitatione motuque corporis excitetut. Iam undique silvae et solitudo ipsumque illud silentium.
    - (1) Non est quod contemnas hoc studendi genus.

## Create the EDD

Analyzing the outline, you can see that it needs just a small number of elements in the EDD.

The EDD defines the elements, containment rules for each element, and the formatting for their text. The element list and their containment rules are:

- Outline (the root element—contains a Title, followed by Author, then followed by the outline Body)  
General Rule: Title, Author, Body
- Body (contains any number of outline items)  
General Rule: item\*
- Title  
General Rule: <TEXT>
- Author  
General Rule: <TEXT>
- p (for paragraphs within each item)  
General Rule: <TEXT>
- item (contains the item text, followed by any number of paragraphs, then followed by an number of other items)  
General Rule: <TEXT>, p\*, item\*

In addition, the EDD specifies paragraph formatting for the outline. This is most important for the outline items and paragraphs because that is what determines item indents and numbering. The EDD uses the same paragraph formats that exist in the template, and uses format rules to determine how to apply those formats according to item nesting:

A format rule to assign paragraph formats according to nested items

**Element (Container):** item

**General rule:** <TEXT>,p\*,item\*

### Text format rules

#### 1. Count ancestors named: item

If level is: 1

Use paragraph format: 1level

Else, if level is: 2

Use paragraph format: 2level

Else, if level is: 3

Use paragraph format: 3level

Else, if level is: 4

Use paragraph format: 4level

Else, if level is: 5

Use paragraph format: 5level

Else, if level is: 6

Use paragraph format: 6level

Else, if level is: 7

Use paragraph format: 7level

Before you convert the outline to structure, you will import the EDD's element definitions into your outline document.

## Generate the Default Conversion Table

The next step is to create the conversion table. With the outline document active, choose *StructureTools>Generate Conversion Table*. The result is a default conversion table:

Default conversion table for an outline

Wrap this object or objects	In this element	With this qualifier
P:Title	Title	
P:Author	Author	
P:1Level	1Level	
P:2Level	2Level	
P:3Level	3Level	
P:3LevelContinued	3LevelContinued	
P:4Level	4Level	
P:4LevelContinued	4LevelContinued	
P:5Level	5Level	
P:6Level	6Level	
P:7Level	7Level	
P:5LevelContinued	5LevelContinued	
P:6LevelContinued	6LevelContinued	

If you were to run this conversion table on the outline, you would get the following results, which are obviously incorrect:

**Edit the Conversion Table**

To generate correct structure for the outline, you have to specify that the outline paragraphs get wrapped in the elements in your EDD. You also have to specify containment rules to tell FrameMaker to properly nest the elements as it converts the document.

If you look at the format catalog in the outline document, or if you look at the default entries in the generated conversion table, you will see that each level of the outline is determined by the paragraph format: 1Level, 2Level, 3Level, and so on. For the paragraphs contained by each outline item, there is a continued format: 3LevelContinued, 4LevelContinued, and so on. But the EDD declares only a single item element and a single

p element. We need to find a way to map all the outline items into these item and p elements in such a way that they capture the nested outline structure, and FrameMaker can automatically apply the correct formats (indenting and numbering).

To do this, we will use qualifiers in the conversion table that identify the level for each item and p. In addition, we will use containment rules to specify what each level of an item can contain. The final conversion table looks like the following:

The edited conversion table

Wrap this object or objects	In this element	With this qualifier
P:7Level	item	7
P:6LevelContinued	p	6
P:6Level,E:p[6]*,E:item[7]*	item	6
P:5LevelContinued	p	5
P:5Level,E:p[5]*,E:item[6]*	item	5
P:4LevelContinued	p	4
P:4Level,E:p[4]*,E:item[5]*	item	4
P:3LevelContinued	p	3
P:3Level,E:p[3]*,E:item[4]*	item	3
P:2Level,E:item[3]*	item	2
P:1Level,E:item[2]*	item	1
E:item[1]*	Body	
P:Title	Title	
P:Author	Author	
E:Title,E:Author,E:Body	Outline	

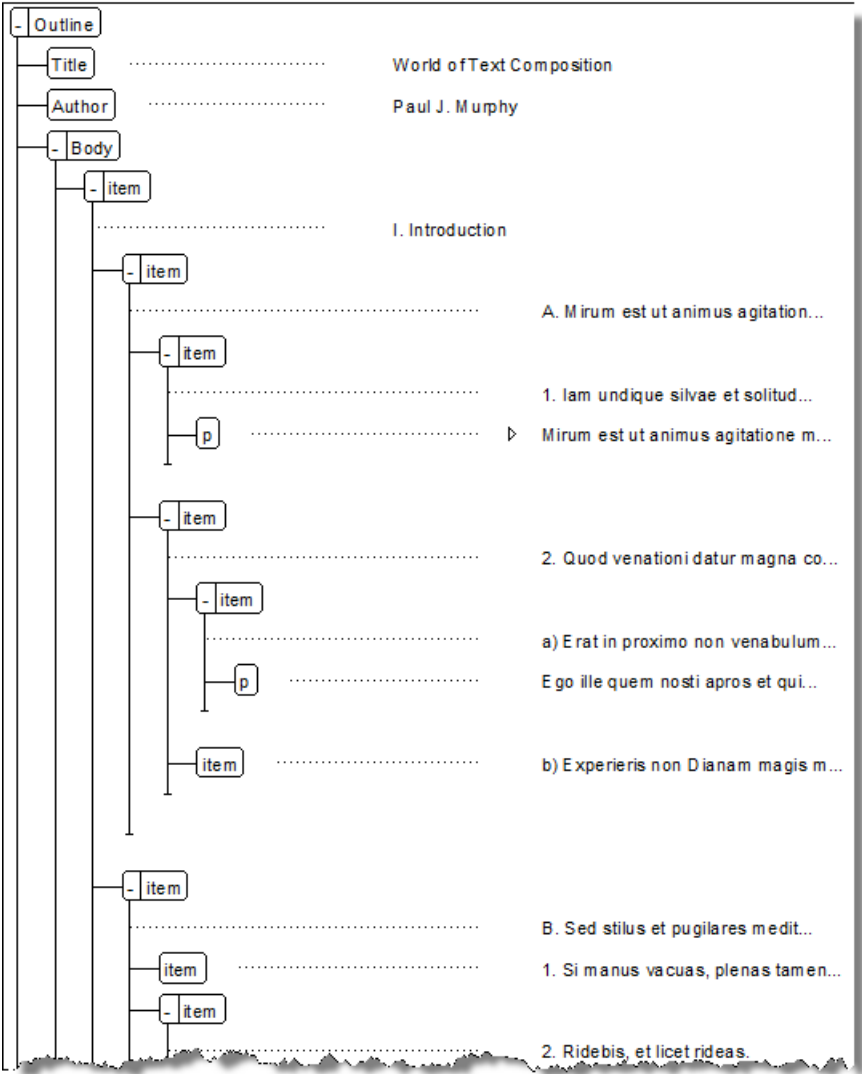
The table works as follows: For each 7Level paragraph, wrap the paragraph in an item, and mark it as a 7 (for the level). For each 6LevelContinued paragraph, wrap it in a p and mark it as a 6. So far this is easy to follow.

Now, for a 6Level paragraph, we want to construct an outline item that could possibly contain any number of 7-level items. The 6Level rule says, wrap a 6Level paragraph in an item marked 6. If the 6Level paragraph is followed by one or more p elements marked as 6, include those in the item. And if the last p[6] element is followed by one or more item[7] elements, include those in the item[6] as well. This rule constructs 6-level items with all their content properly structured.

For the rest of the outline levels, you can see that the rules are pretty much the same. Then, for one or more item[1] elements, wrap them all in a Body element. The Body element is almost at the top level of the hierarchy. The conversion table also creates Title and Author elements at the same level as Body. When all this is done, the conversion table wraps everything up in the Outline element, which is the document root.



The result of running this conversion is as follows:



Now you have a conversion table that works with any outlines that use the template correctly. You can use this to convert a collection of unstructured outlines to XML. First run the conversion on your outlines, and then save the structured outlines as XML.

TIP: You can use FrameMaker ExtendScript to make a batch process that saves a collection of structured files as XML.

### Conclusion

Unless you are setting up a publication environment from scratch, it's very likely that you will have legacy unstructured documents. If it's worth the effort to create an XML authoring environment, then it is probably just as worthwhile to convert your legacy to XML.

In many organizations, the requirement is to implement XML authoring for existing unstructured publications. In that case, you should consider running a pilot project first, as a proof of concept. Begin with analysis and create an EDD. Then try to manually reproduce a representative document using that EDD. This is a good way to verify that the EDD supports the document constructs that you need.

Once you have verified the EDD, you can begin building the conversion table. This is usually an iterative process, where you convert a portion of a document, check the results, then move on to convert more.

It's possible that some of your legacy constructs are so complicated that it's difficult to build a working conversion. A good trick is to build a conversion table that handles most of your document, and then you can use FrameMaker ExtendScript to automatically find and fix any invalid structure.

Remember the 80/20 rule: It takes 20% of the time to automate 80% of your work. But to automate the last 20% of your work will take 80% of the time. If you have hit a wall with the conversion table, maybe ExtendScript can take over. If even that is too difficult to program, then you can search for invalid structure in your document and fix it manually. The amount of manual fixes you need to make should be very small. Remember, converting a body of documents to structure is a one-time effort.

## Where To Go From Here

As you move into structured authoring, you may want consult some of the resources listed in this section.

### Adobe White Papers

Visit our FrameMaker site ([www.adobe.com/products/framemaker/](http://www.adobe.com/products/framemaker/)) for links to additional white papers, customer success stories, and more.

### Recorded Adobe Webinars

At the following website, you will find many recorded Adobe Tech Comm webinars that have XML or DITA as a theme:

<http://adobe.ly/Pbdp0J> ... just search for "XML" or "DITA".

You will also find the following blog to be of great interest:

"Classic 7-Part Aldous series on Unstructured to XML webinar series republished"

This blog has links to a seven-part recorded webinar series on all the steps required to migrate from unstructured to structured or XML documents in FrameMaker. Although this webinar was created for an earlier version, it is still highly relevant for the latest release of FrameMaker. You can find this blog at:

<http://blogs.adobe.com/techcomm/2013/06/classic-7-part-aldots-series-on-unstructured-to-xml-webinar-series-republished.html>

### Structure Developer's Guide

The "Structure Developer's Guide," found in the OnlineManuals folder of the FrameMaker installation directory, provides detailed information about building structured applications in FrameMaker. It also contains a complete read/write rules reference.

### DITA

You can find more information about the Darwin Information Typing Architecture at this site:

[www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=dita](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita)

### Docbook

For more information about DocBook, visit the official DocBook site:

[www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=docbook](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=docbook)

## Training

You can search our site for an authorized training provider in your area. Vendors offer scheduled public classes with open enrollment and private classes at your site or theirs. Some vendors also offer distance learning or self-study materials. You can search for an Adobe authorized training provider [here](#):

**Alternatively, for any queries, please reach out to us at [techcomm@adobe.com](mailto:techcomm@adobe.com)**

## Consulting

Adobe maintains a list of FrameMaker consultants. Please reach out to Adobe Technical Communication at [techcomm@adobe.com](mailto:techcomm@adobe.com) for consulting references. You can also find qualified consultants by searching for them on the web. In addition, many authorized training providers also provide consulting services.



**Adobe**

Adobe Systems Incorporated  
345 Park Avenue  
San Jose, CA 95110-2704  
USA  
[www.adobe.com](http://www.adobe.com)

Adobe, the Adobe logo, 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.