Migrating from Unstructured to Structured FrameMaker

TABLE OF CONTENTS

Implementing structured Adobe® FrameMaker® is both an opportunity and a challenge. You can combine FrameMaker's powerful features with structured authoring to create a sophisticated workflow. Within this new authoring environment, you can automate formatting, create, edit, and publish XML-based content, and enforce required document structures.

The challenge is that building a structured workflow requires you to master new tools (such as the structure features in FrameMaker), new concepts (structured authoring), and new technology (XML, structure definitions, and perhaps Extensible Stylesheet Language (XSL) transformation). If you have no immediate need for XML, you can implement structure in FrameMaker without providing support for XML import or export. This approach has a smaller learning curve and is discussed in detail later in this document.

This white paper assumes that you are already working in unstructured (paragraph-based) FrameMaker and are considering a move to structured FrameMaker.

What is XML?

Extensible Markup Language (XML) is a vendor-neutral, open format managed by the World Wide Web Consortium (W3C). Like HTML, XML consists of tags enclosed in angle brackets (such as <*example*>), is relatively easy to learn, and is stored in a text file. Aside from their superficial similarities, HTML and XML are used very differently.

HTML coding is often done on the fly, but XML requires a rigorous approach. A fundamental rule for an XML file is that it must be *well-formed,* which means that the XML document follows XML syntax rules. For example, all tags must be opened and closed properly:

```
<p>This version is correct.</p><p>Each paragraph is opened and closed.</p>

<p>This is not allowed.<p>The first paragraph must be closed before the second paragraph starts.
```

Well-formed documents are predictable, which makes it easier for software to process the XML files automatically.

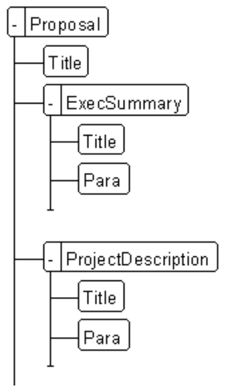
Note: *It's possible to create well-formed HTML, but it's not required in most environments. Web browsers can usually render HTML that is not well-formed. XML must always be well-formed.*

XML lets you define tags, thus creating your own markup language to describe document content. For example, in a proposal document, you could create tags such as ProjectDescription and Costs.

The XML file is human-readable, but it can be tedious to identify where each tag begins and ends. In the following example, the tags are bolded to make them easier to find.

```
<?xml version="1.0" encoding="utf-8"?>
<Proposal>
  <Title>Proposal for Widget Production</Title>
  <ExecSummary>
    <Title>Executive Summary</Title>
    <Para>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ultrices.</Para>
  </ExecSummary>
  <ProjectDescription>
    <Title>Project Description</Title>
    <Para>Duis vel libero. Integer venenatis nibh non magna. Morbi id leo.</Para>
  </ProjectDescription>. . .
</Proposal>
```

Structured FrameMaker gives you a tree view of the document that arranges the elements hierarchically. It's easy to see the relationships among the tags.

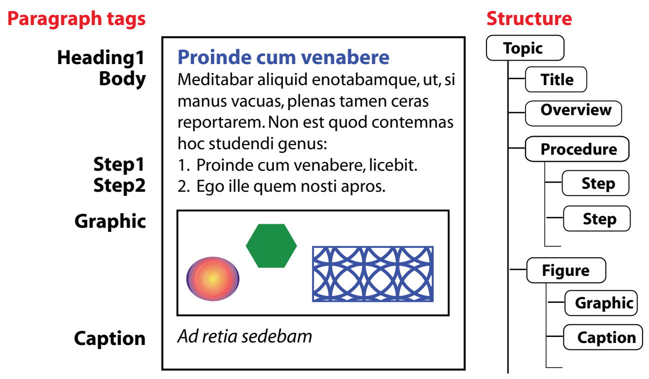


FrameMaker renders the XML tags in a tree view

Capturing information hierarchy

Instead of style-based, paragraph-oriented word processing and desktop publishing, XML provides a foundation for *structured authoring;* content is described by elements that are organized in a hierarchical tree. In word-processing environments (such as unstructured FrameMaker), the relationship among the various document components is apparent through formatting on the page. The document file, however, does not capture these relationships because a word processor document is made up of a string of paragraphs. The relationship between a Body paragraph tag and the preceding Heading1—the Body tag is logically subordinate to the Heading1—is not represented in an unstructured FrameMaker file. Structured authoring, however, does capture the hierarchical relationships among the document components.

The following figure shows unstructured and structured representations of the same document. The unstructured paragraph tags are in a flat list; the structured elements are grouped to show relationships. The Graphic and Caption element, for example, share a common Figure parent. Step elements are subordinate to the Procedure element.


Paragraph tags
Heading1
Body
Step1
Step2
Graphic
Caption
Proinde cum venabere
Meditabar aliquid enotabamque, ut, si manus vacuas, plenas tamen ceras reportarem. Non est quod contemnas hoc studendi genus:
1. Proinde cum venabere, licebit.
2. Ego ille quem nosti apros.
Ad retia sedebam
Structure
Topic
Title
Overview
Procedure
Step
Step
Figure
Graphic
Caption

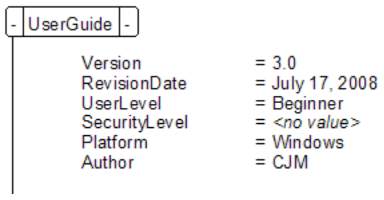
Word processing versus structured authoring

Embedding metadata in XML

Elements in XML can have attributes, which enables you to store additional information about an element. For example, you could provide an attribute that specifies the classification level for a topic:

```
<topic classification="secret">Secret information here</topic>
```

Often, attributes are used for basic document metadata, such as the name of the author and the last revision date of a particular document.



```
UserGuide
  Version        = 3.0
  RevisionDate   = July 17, 2008
  UserLevel      = Beginner
  SecurityLevel  = <no value>
  Platform       = Windows
  Author         = CJM
```

Embedding document metadata

Note: *For a more detailed overview of XML, refer to www.adobe.com/products/framemaker/pdfs/xml_fm7.pdf.*

Why XML?

Creating an XML-based workflow is a significant effort, but it opens up some exciting new opportunities, such as improving the consistency and organization of your files, reusing content, and reducing publishing costs. The following sections provide some details.

Enforcing consistent organization

In unstructured FrameMaker, a template specifies which formatting components are available in your files. Along with the template, you provide usage rules. For example, you might specify that a simple proposal should have the following content sequence:

1 Proposal title

2 Executive summary

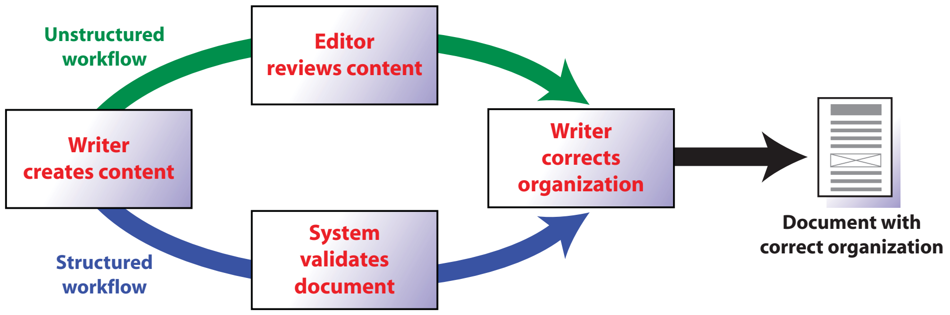
3 Project description

4 Schedule

5 Cost

You cannot enforce this required sequence in an unstructured template. Instead, you create a style guide, which authors are supposed to read and comply with. To verify that a particular document follows the rules in the style guide, a human editor must read the proposal and verify that each section is in the proper order.

XML technology lets you describe the required organization of the content with structure rules. A file that follows these rules is *valid*. You can use software to validate your file and verify that it conforms to the rules you've established, so that instead of a human reading the file to check its organization, the software can do the validation work.


Unstructured workflow
Editor reviews content
Writer creates content
Writer corrects organization
System validates document
Structured workflow
Document with correct organization

Structured and unstructured workflow to create a document

In addition to built-in validation features, structured FrameMaker also provides *guided editing*, which supports authors by displaying only the allowed elements as they are working. In other words, you can now enforce your document organization through the template (rather than by human review).

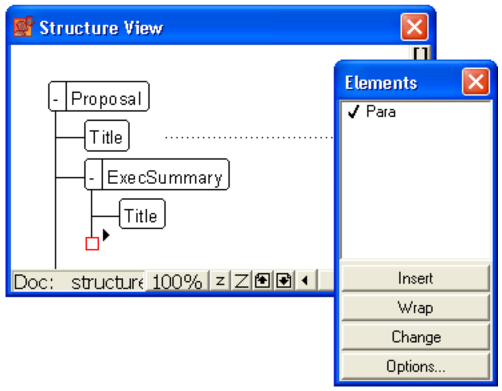
For a proposal document, the structure rule would read as follows:

```
Title,ExecSummary,ProjectDescription,Cost,Schedule
```

This rule indicates that a syntax reference section *must* contain one Title, followed by one ExecSummary, and so on. You can then specify further structure inside each element. For example, you can require that the ExecSummary element contains a title and a paragraph:

```
Title,Para
```

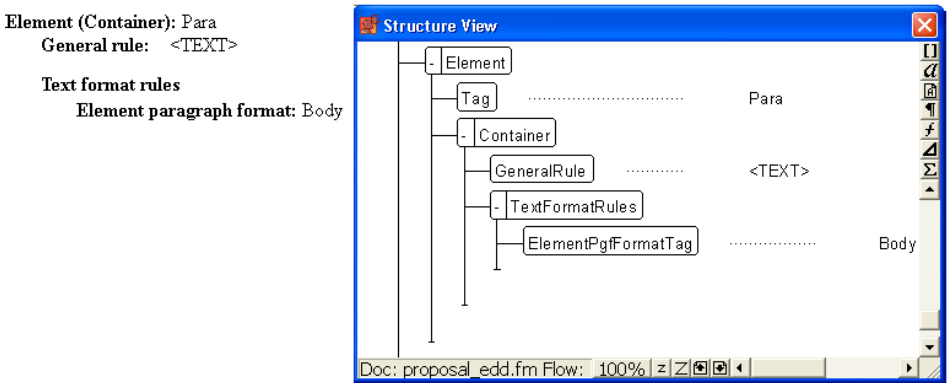
A document that omits one of these items, or that uses a different order for the elements, is not allowed. FrameMaker's structure view indicates in real-time where required content is missing, as shown in the following figure.



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.

Automating and enforcing formatting

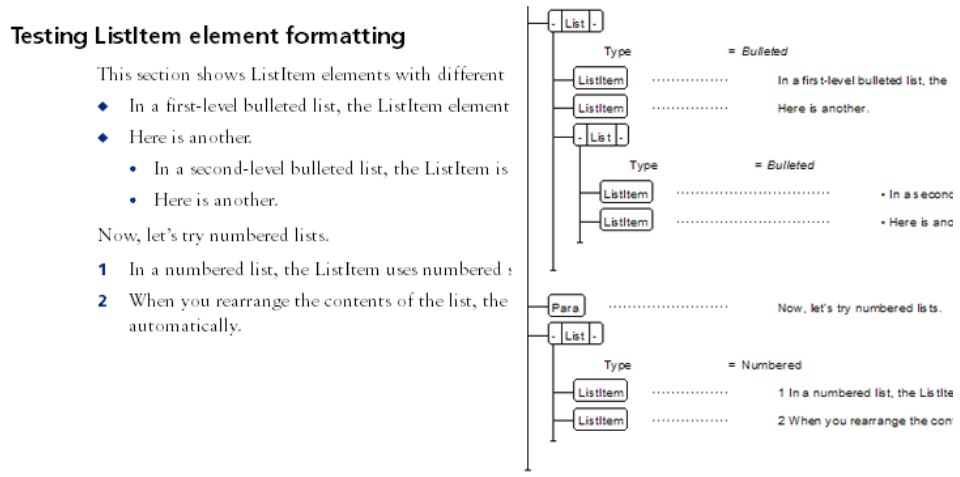
In structured FrameMaker, authors do not work with formatting components such as paragraph and character tags. Instead, they insert elements and element content into their documents. Document formatting is applied automatically based on formatting information that is embedded in the structured template. The structured template is made up of two components: formatting information and structure definitions. The structure definitions are stored in an element definition document (EDD), which in turn references formatting information.


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

Formatting specifications in structured FrameMaker

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.*


Testing ListItem element formatting

This section shows ListItem elements with different

- In a first-level bulleted list, the ListItem element
- Here is another.
  - In a second-level bulleted list, the ListItem is
  - Here is another.

Now, let's try numbered lists.

1 In a numbered list, the ListItem uses numbered

2 When you rearrange the contents of the list, the automatically.

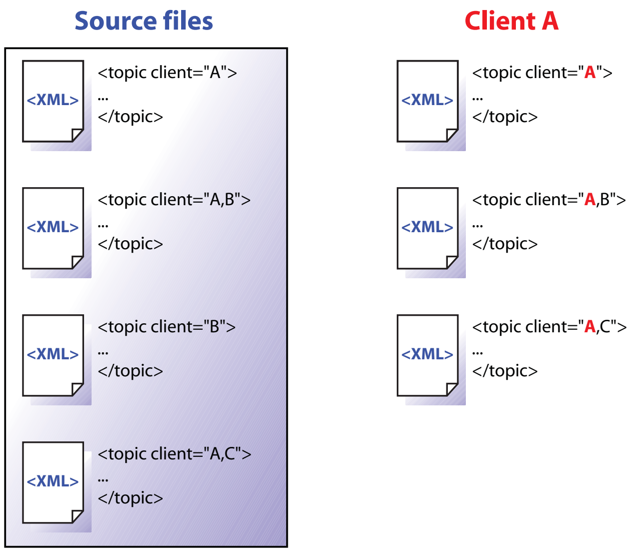
```
List
  Type = Bulleted
  ListItem ........ In a first-level bulleted list, the
  ListItem ........ Here is another.
  List
    Type = Bulleted
    ListItem ........ • In a seconc
    ListItem ........ • Here is anc
Para ........ Now, let's try numbered lists.
List
  Type = Numbered
  ListItem ........ 1 In a numbered list, the ListIte
  ListItem ........ 2 When you rearrange the con
```

ListItem formatting in different contexts

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

Using metadata for versioning

At the document level, metadata provides keywords and other descriptions of a document. You can also use metadata to identify information that is unique for a particular version of a deliverable. This technique lets you use a single set of source documents to deliver highly customized information for each version. Consider, for example, the problem of customized software development. If your company customizes its products for each client, you need a way to identify what information belongs to each client. 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 deliverables.

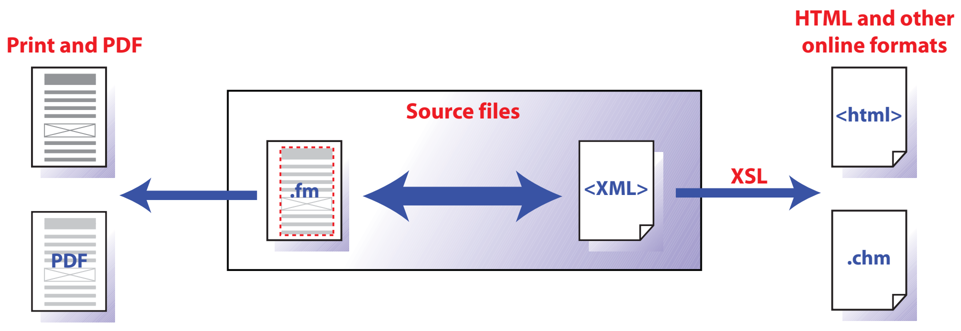


```
Source files
<topic client="A">
...
</topic>

<topic client="A,B">
...
</topic>

<topic client="B">
...
</topic>

<topic client="A,C">
...
</topic>

Client A
<topic client="A">
...
</topic>

<topic client="A,B">
...
</topic>

<topic client="A,C">
...
</topic>
```

Filtering to produce client-specific deliverables

Publishing to multiple output formats

Many FrameMaker users are already familiar with single-sourcing workflows. In a structured FrameMaker environment, you can take advantage of FrameMaker's powerful print and PDF capabilities, and then generate XML to create other outputs, such as HTML and online help formats.

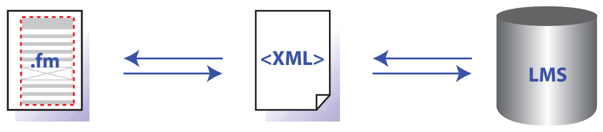


Structured FrameMaker and XML in a multichannel workflow

You can also continue to use single-sourcing workflows that are based on unstructured conversion tools.

Supporting content reuse and exchange with XML

XML lets you move information from one authoring tool to another without losing information. You can develop information in FrameMaker, save it out to XML, and then open the XML files in another tool. Perhaps a technical writing group (which uses FrameMaker) needs to share information with a training group (which uses an XML-based learning management system). You can use XML to exchange content between two otherwise incompatible applications.



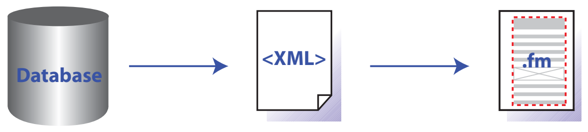
Reusing content between FrameMaker and a learning management system (LMS)

Reducing localization cost

When content is localized, the desktop publishing component is a significant part of the cost. 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

Instead of connecting directly into a database, you can use XML as an intermediate format. From the database, you generate an XML file that contains the information you want to publish. On the FrameMaker side, you set up an import application that provides formatting information for the XML elements. You then open the XML file in FrameMaker, and the database information is formatted automatically.



Database publishing with FrameMaker and XML

For more information about database publishing, refer to the "Adobe FrameMaker 7.1 Database Publishing" at www.adobe.com/products/framemaker/pdfs/database_fm7.pdf.

Note: *To enable you to build automated, server-based solutions, Adobe offers the FrameMaker Server license. For more information, visit the FrameMaker Server page, www.adobe.com/products/fmserver/main.html.*

Complying with required document structures

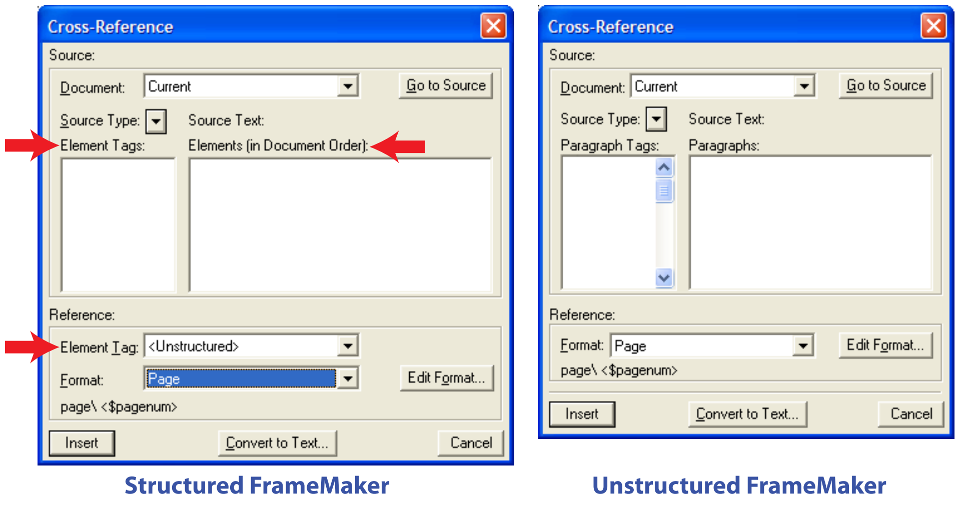
The ability to enforce structure based on explicit rules improves consistency for any document. In regulated industries, the ability to verify that the required structures are being followed is critical. If your organization is required to comply with specific structures due to government or industry standards, structured authoring can help you ensure that your documents conform.

Why structured FrameMaker?

Structured FrameMaker produces a user-friendly structured authoring environment that lets you create XML. Authors making the transition from unstructured FrameMaker will find many familiar publishing features. Both versions of FrameMaker excel in creating, editing, and producing complex content, so authors can easily create PDF files with bookmarks and hyperlinks, 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.

Migrating from unstructured FrameMaker to structured FrameMaker

The transition from unstructured, paragraph-based authoring to structured, element-based authoring can be challenging. Authors must shift from the desktop publishing model to working with a tree view of a document. Moving authors who are familiar with unstructured FrameMaker into structured FrameMaker reduces the learning curve. Although the authoring experience changes, knowledge from unstructured FrameMaker is helpful when working in the structured environment. For example, authors still insert cross-references with the Cross-Reference dialog box, but instead of choosing paragraph tags and paragraph instances, they choose elements and element instances.

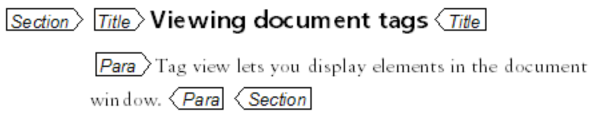


Knowledge of cross-references from unstructured FrameMaker is helpful in structured FrameMaker

If you are working in unstructured FrameMaker, it's likely that your documents follow a template with minimal formatting overrides. This document consistency makes it easier to convert to structured documents. If, however, authors create content with no accountability for consistent, repeatable formatting, then document conversion will be a challenge. Most authors in FrameMaker-based environments are accustomed to following template rules, and that discipline makes the transition to structured authoring easier.

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. Inside the document window, you have the option to display element tags for a structured document.



Section Title Viewing document tags Title

Para Tag view lets you display elements in the document window. Para Section

Viewing element boundaries as tags

Creating 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 a layout for print and PDF is much more complex than publishing to the web. FrameMaker supports printing for structured documents and also lets you create PDF files with bookmarks automatically built from a document's element or paragraph tagging.

Relatively low licensing cost

FrameMaker 7 gives you everything you need to build a structured authoring environment. You can build and configure the environment before you deploy it to other authors. This gives you the ability to create a structured environment without making a huge commitment to new software. You can scale the implementation by rolling out FrameMaker and your structure configuration files to additional authors as needed.

Relatively low implementation effort

Implementing any structured authoring environment requires a significant effort. Reusing an existing formatting template reduces the implementation effort required. FrameMaker also provides tools to help you convert your unstructured documents into structured documents.

XML implementation options

You can combine structured FrameMaker and XML in several different ways. FrameMaker does not lock you into a single workflow. Your options include the following:

- *FrameMaker authoring with XML support.* With this option, you use FrameMaker as your main authoring tool. As needed, you save your files from FrameMaker out to XML, and then use that XML as needed. All content changes are made in FrameMaker; when you need new XML files, you save out to XML again.
- *FrameMaker authoring with XML source files.* With this option, you use FrameMaker as your main authoring tool. Your source files are XML. You open the XML files in FrameMaker, make changes, and save. When you save, your XML source file is updated.
- *FrameMaker and XML authoring.* With this option, you produce content in FrameMaker and in other XML tools. When you are ready to publish to print, you combine the XML created outside FrameMaker with FrameMaker-based content, and then produce your final print or PDF deliverable. This workflow is useful in an environment with many occasional content contributors. The content contributors may not need a powerful publishing environment. Therefore, they may choose to produce their content in a low-end XML editor, and then deliver the XML files to the FrameMaker-based publishing team. This workflow is also relevant if you have XML generated from a database or another line-of-business application.
- *XML authoring with FrameMaker publishing.* With this option, you work mainly in another XML authoring tool, and XML content is produced by a database system or other line-of-business application, or you are receiving XML from outside your organization. When content is finished, you produce print and PDF by bringing the XML into FrameMaker. You might build a server-based application with FrameMaker Server to automate this process.

Components of a structured authoring environment

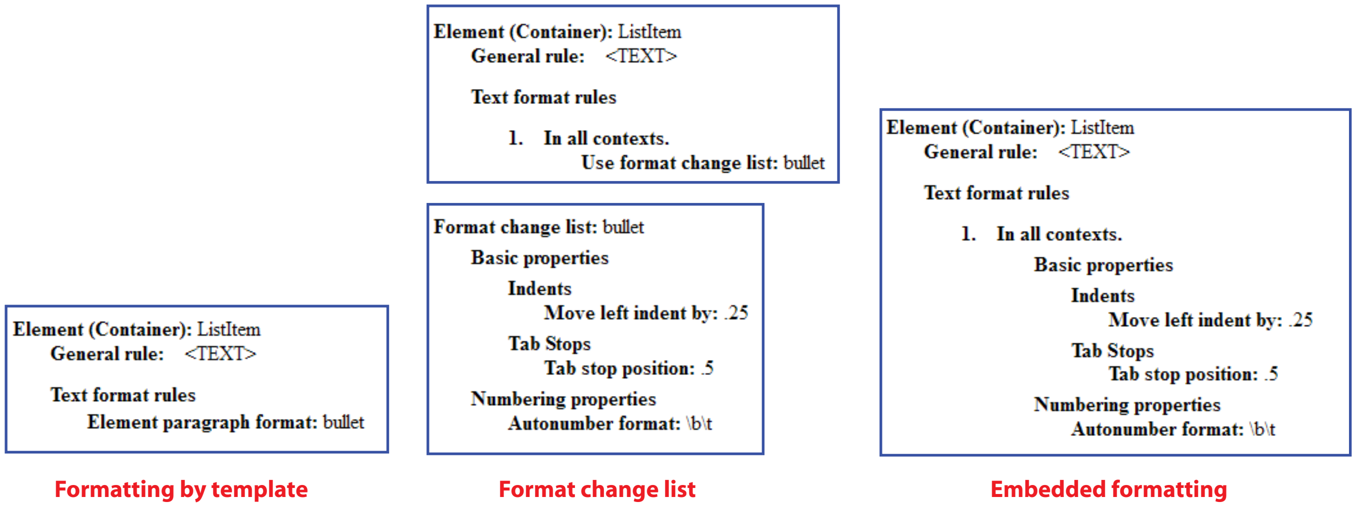
In unstructured FrameMaker, setting up an authoring environment requires only a template file. Once you define the template and distribute it, authors are ready to begin work. In structured FrameMaker, you must provide some additional files to authors.

Authoring only in structured FrameMaker

You can implement structure in FrameMaker without providing support for XML import or export. If you have no immediate need for XML, this approach lets you experiment with structured authoring.

To establish a basic structured authoring environment in FrameMaker, you must create an element definition document (EDD) that specifies 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 information in that template.
- *Format change lists.* You can create named formatting definitions in the EDD. The format change lists lets you provide partial formatting specifications and inherit the rest from a few base paragraph tags. You can reuse format change lists for related elements.
- *Embedded formatting.* You can specify formatting by writing the formatting into each element definition.



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

  Text format rules
    Element paragraph format: bullet

Formatting by template

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

  Text format rules

    1. In all contexts.
      Use format change list: bullet

Format change list: bullet
  Basic properties
    Indents
      Move left indent by: .25
    Tab Stops
      Tab stop position: .5
  Numbering properties
    Autonumber format: \b\t

Format change list

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

  Text format rules

    1. In all contexts.
      Basic properties
        Indents
          Move left indent by: .25
        Tab Stops
          Tab stop position: .5
      Numbering properties
        Autonumber format: \b\t

Embedded formatting

Comparing three formatting options in the EDD

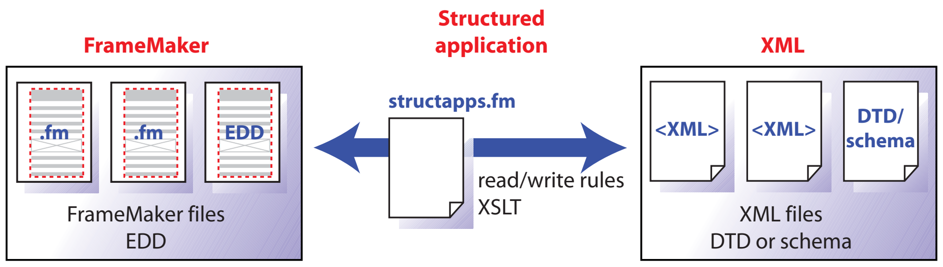
If you choose to use format change lists or embedded formatting, you will still need a basic formatting template that defines your master pages.

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

Structured authoring and XML

To provide support for XML import and export, you need the EDD, structured template, and a few additional 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.* The read/write rules control translation from FrameMaker structure to XML and back. Specifically, read/write rules help manage table translation and other complex formatting.
- *Structured application.* The structured application lists all of the component files in your structure implementation and provides configuration information. For example, you can use the structured application to specify whether and how to export conditional text tagging (version 7.1 and higher).



Components of a structured authoring environment

EDDs, XML, and SGML

FrameMaker EDDs support both XML and the older, more complex, SGML standard. A few EDD features are provided for SGML compatibility but should be not used in EDDs for XML. If you are designing an EDD for use with XML, do not use any of the following:

- Inclusions and exclusions
- Ampersand (&) connector
- Element names that begin with numbers
- Special characters or whitespace characters (space, tab) in element names

Extras

If the requirements for your authoring environment go beyond what's available with a basic structured application, FrameMaker offers several additional ways to customize XML processing. You can use any or all of the following:

- *XSL pre- and post-processing* (new in version 7.2). In addition to processing content with read/write rules, you can supply XSL transformations that process your XML files during import or export. For example, you can rearrange the order of elements in your XML files during import with an XSL transformation.

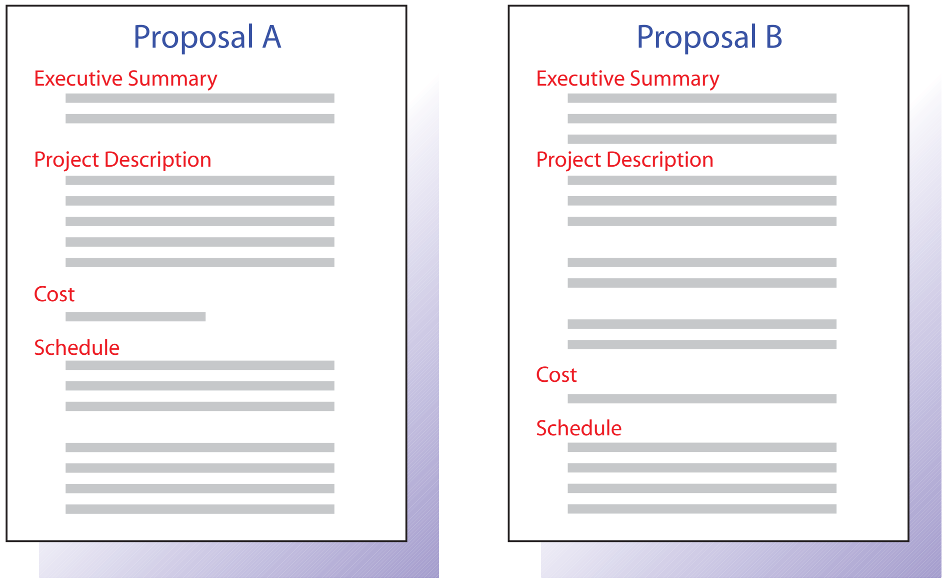
- *FrameMaker API client.* The FrameMaker Developer's Kit (FDK, available at partners.adobe.com/public/developer/framemaker/devcenter.html) allows you to customize and extend FrameMaker's functionality. For example, you could use the FDK to change how index markers are processed.
- *Third-party plug-ins.* Numerous plug-ins are available that extend FrameMaker's functionality. Here are some interesting possibilities:
  - Writing scripts to automate FrameMaker (FrameScript, www.framescript.com).
  - Showing and hiding information based on attribute values (Sourcerer, www.advantica.biz/sourcerer/).
  - Exporting content to HTML and online help formats (WebWorks Publisher Professional Edition, www.quadralay.com, and MIF2GO, www.omsys.com).
  - Providing content management. Most content management systems support XML files; a few also support FrameMaker files natively.

Getting started with structure

We recommend that you start by building a small structured application to familiarize yourself with all the components you need. This section describes an application for a simple proposal.

Content analysis

Before building the proposal template, analyze existing proposals to identify their components.



Reviewing existing proposals

Based on this analysis, you create a content map. The proposal example results in the following sequence:

- Title
- Executive summary
  - Title
  - One paragraph
- Project description
  - Title
  - One or more paragraphs

- Cost
  - Title
  - One or more paragraphs
- Schedule
  - Title
  - One or more paragraphs

***Note:** You could probably build the EDD for this simple example without formal content analysis. For larger projects, though, content analysis is critical.*

Choosing an EDD strategy

There are several ways to begin building your EDD. You can:

- Create the entire EDD yourself.
- Import a DTD or Schema to create an EDD that contains structure definitions.
- *(version 7.2)* Use a conversion rules table to structure an existing sample document and create a first draft of the EDD that contains basic element definitions and formatting that matches your unstructured template.
- Modify an existing EDD—either one of the samples supplied with FrameMaker or an EDD from another source.

This document describes how to create a new EDD so that you can see how the process works.

Building the proposal EDD

Based on the content analysis, you can now create the proposal EDD.

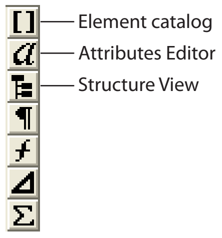
To build the EDD:

1 Make sure you are in structured FrameMaker. To switch from unstructured to structured FrameMaker, select File > Preferences > General. In the Product Interface drop-down list, select Structured FrameMaker. Close and restart FrameMaker.

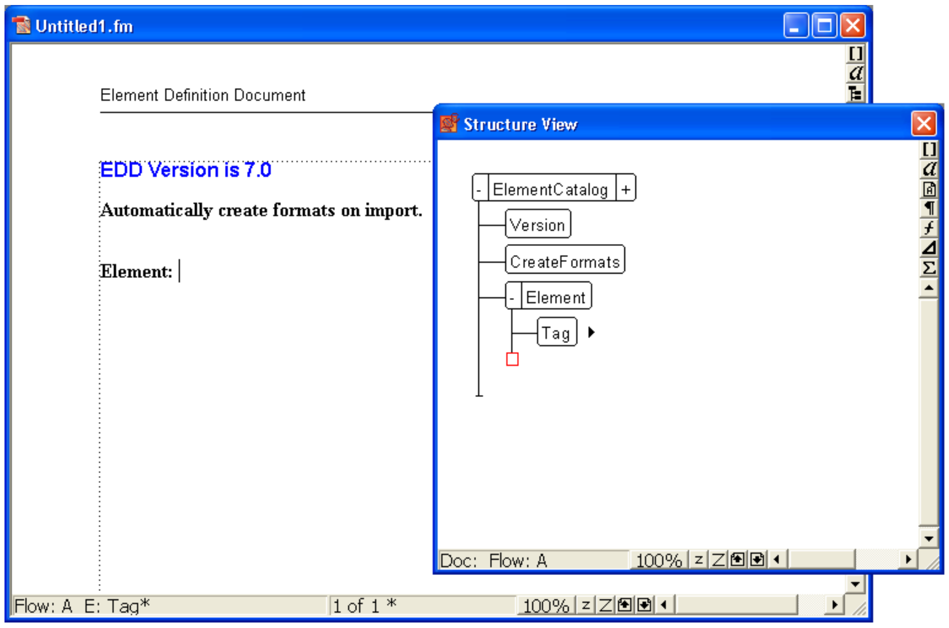
2 Select File > Structure Tools > New EDD to create a new EDD file. FrameMaker inserts the first few default elements in the EDD for you.

***Note:** The EDD is itself a structured FrameMaker document. You use the same guided editing environment to create the EDD that you use to edit other structured documents.*

3 Click the Structure View button to display the Structure View window. (The Structure View button is found in the top right edge of your document window. Refer to the following figure.)



Structure View and other buttons



Element Definition Document

EDD Version is 7.0

Automatically create formats on import.

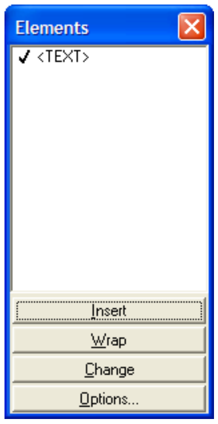
Element:

ElementCatalog
Version
CreateFormats
Element
Tag

Getting started with an EDD

4 Create the top-level Proposal element. Position your cursor to the right of the Tag bubble in the Structure View, and type in `Proposal`. As you type, the letters appear in both the Structure View and the document window.

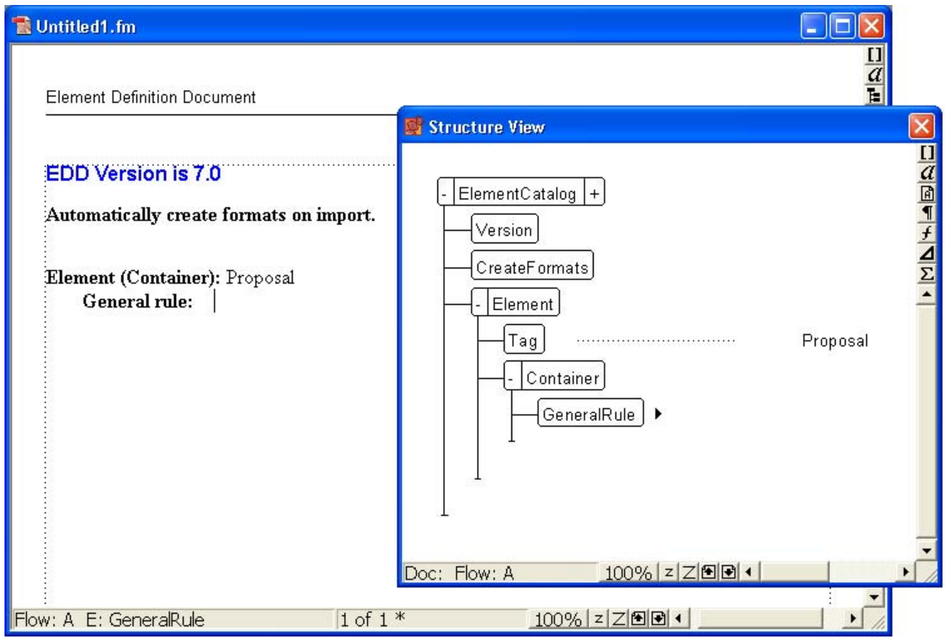
5 Click the Elements Catalog button (found in the top right edge of the document window and of the Structure View) to display the Elements catalog.



Elements catalog

If your Elements catalog doesn't behave as expected, check your element options. Select Element > Set Available Elements to display the Set Available Element dialog box. Select Valid Elements for Working Start to Finish, and then click Set.

6 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).



Element Definition Document

EDD Version is 7.0

Automatically create formats on import.

Element (Container): Proposal
General rule:

Adding the general rule for Proposal

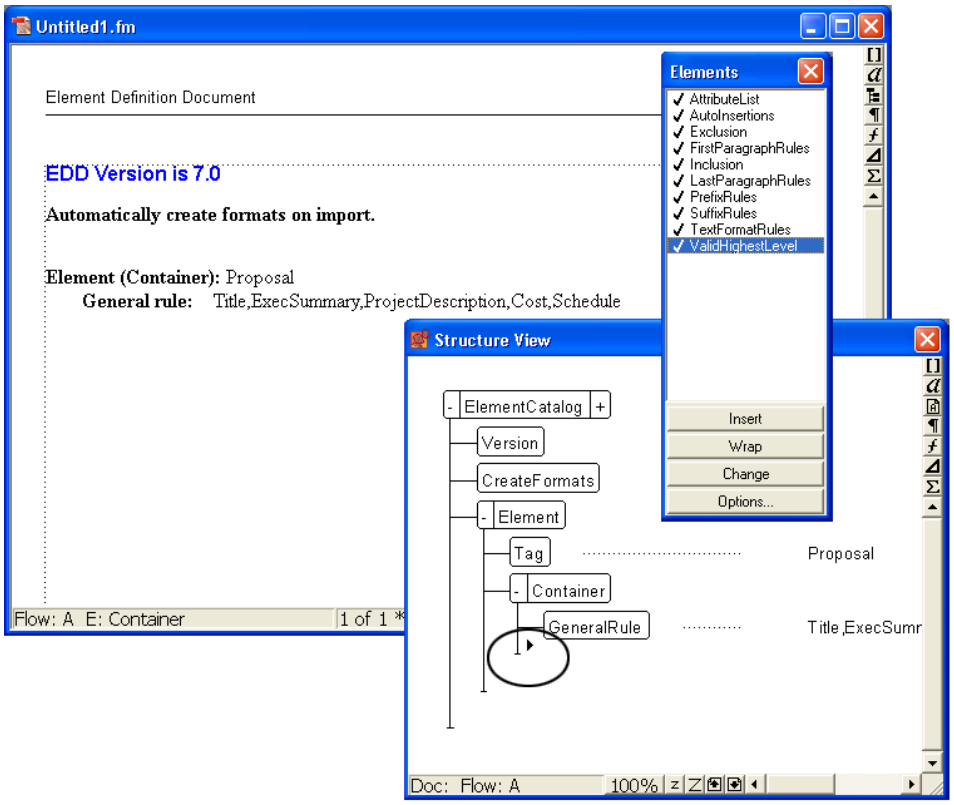
The general rule specifies which elements are allowed inside the proposal element. During the content analysis, you identified the following: title, executive summary, project description, cost, and schedule.

7 Type a general rule for Proposal:

Title, ExecSummary, ProjectDescription, Cost, Schedule

Note: *Element names cannot contain spaces.*

8 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, click the ValidHighestLevel element in the Elements catalog, and then click Insert.



Element Definition Document

EDD Version is 7.0

Automatically create formats on import.

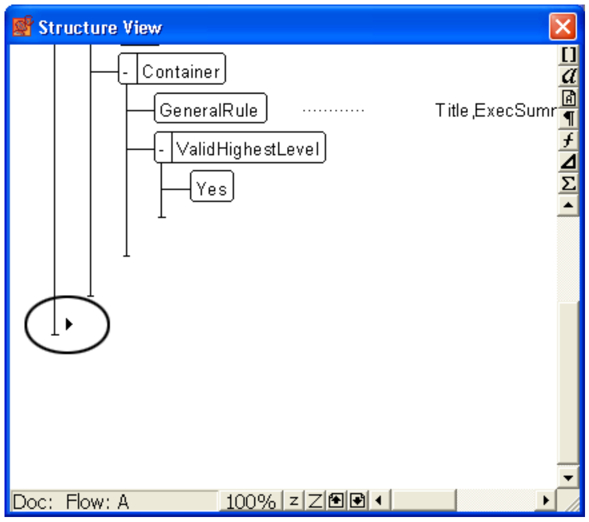
Element (Container): Proposal
General rule: Title,ExecSummary,ProjectDescription,Cost,Schedule

Inserting the ValidHighestLevel element

The Proposal element is complete. You must now provide definitions for each of the child elements: ExecSummary, ProjectDescription, Schedule, and Cost.

To define the child elements:

1 Position your cursor at the bottom of the structure as shown in the following figure.

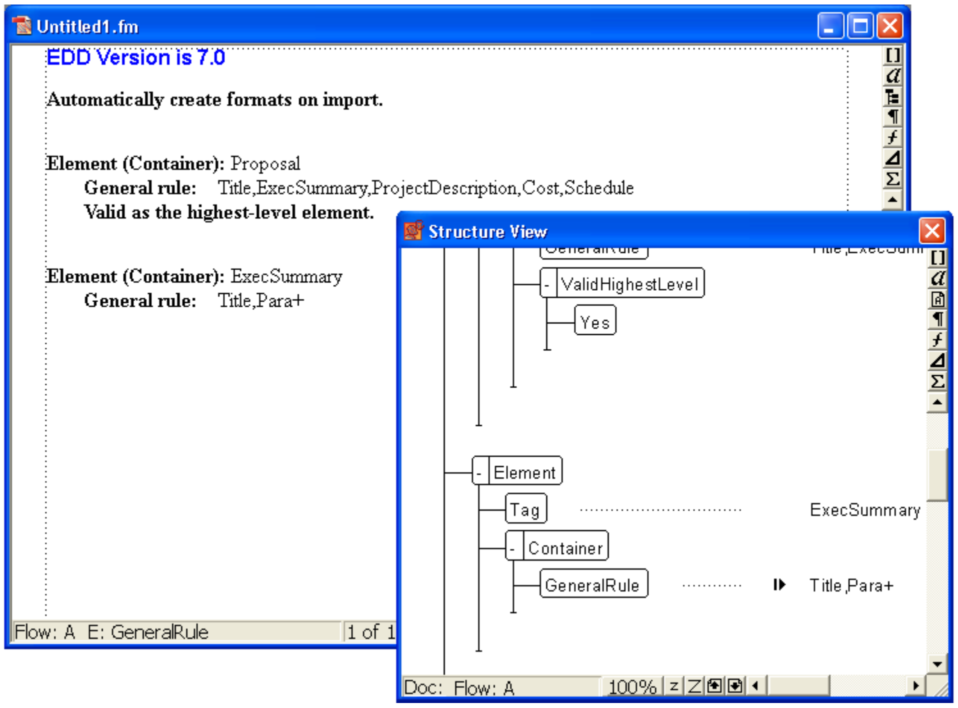


Getting ready to insert another element definition

2 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+

The result is shown in the following figure.



EDD Version is 7.0

Automatically create formats on import.

Element (Container): Proposal
General rule: Title,ExecSummary,ProjectDescription,Cost,Schedule
Valid as the highest-level element.

Element (Container): ExecSummary
General rule: Title,Para+

Adding the ExecSummary element

3 Repeat step 2 to define the remaining elements. The general rules are shown in the following table.

ELEMENT	GENERAL RULE
ProjectDescription	Title,Para+
Cost	Title,Para+
Schedule	Title,Para+
Title	<TEXT> (Type the word TEXT with angle brackets around it.)
Para	<TEXT> (Type the word TEXT with angle brackets around it.)

4 Save your EDD file as proposal_EDD.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.

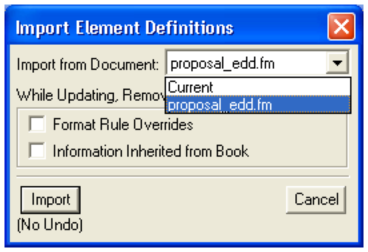
Testing the EDD

To test the EDD, you 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.

To verify that the structure is working as expected:

1 Create a new, blank, portrait document by selecting File > New > Document, and then click Portrait.

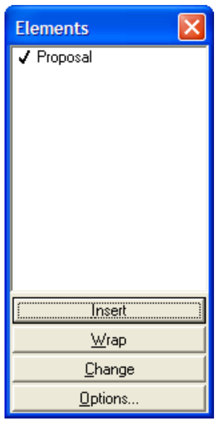
2 Make sure that the EDD and the new document file are both open. From the new document, select File > Import > Element Definitions. In the Import from Document drop-down list, select the proposal EDD and click Import.


Import Element Definitions
Import from Document: proposal_edd.fm
Current
proposal_edd.fm
While Updating, Remov
Format Rule Overrides
Information Inherited from Book
Import
(No Undo)
Cancel

Importing element definitions from the EDD

Note: *If your EDD is not displayed in the list, make sure that the EDD file is open and that you have saved it. Until you save a document, it is not available in this list.*

The structure definitions in your EDD are imported into the blank document. To verify that the definitions were imported, position your cursor in the main text flow and then display the Elements catalog. You should see the Proposal element, as shown in the following figure.

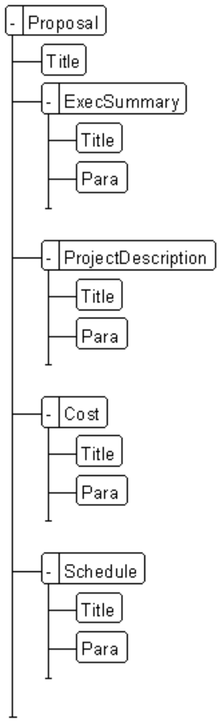


The Elements catalog shows permitted elements

3 Insert a Proposal element. The Title element, which is required as the first child of Proposal, should now appear in the Elements catalog.

4 Insert the Title element.

5 Continue inserting elements until your proposal structure is complete. Your result should look like the figure below.



Completed proposal structure

Note: *If your structure doesn't match the figure above, go back to your EDD, correct it, and then reimport the element definitions. You can also check your EDD against the proposal EDD, which is available at:*

www.adobe.com/products/framemaker/proposal_edd

6 Save your file as structured_proposal.fm.

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. This section describes how to provide formatting, and how to automatically insert the correct text for the various titles.

By default, text uses the Body paragraph tag.

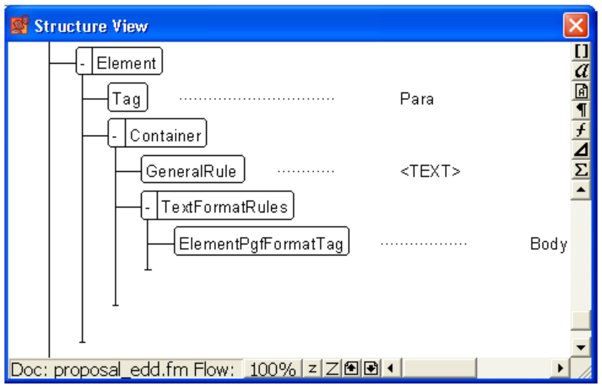
To change the Body tag definition:

1 In structured_proposal.fm, select Format > Paragraph > Designer and change the default definitions of the Body and Heading1 paragraph tags. For example, change the font or put a line above the Heading1. To make your changes obvious, you may also want to assign unique colors to the two tags.

2 In proposal_EDD.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 ElementPgfFormatTag element. Type `Body` as the text for the ElementPgfFormatTag element.

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

Text format rules
Element paragraph format: Body



Adding formatting rules to the EDD

For the Title 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.

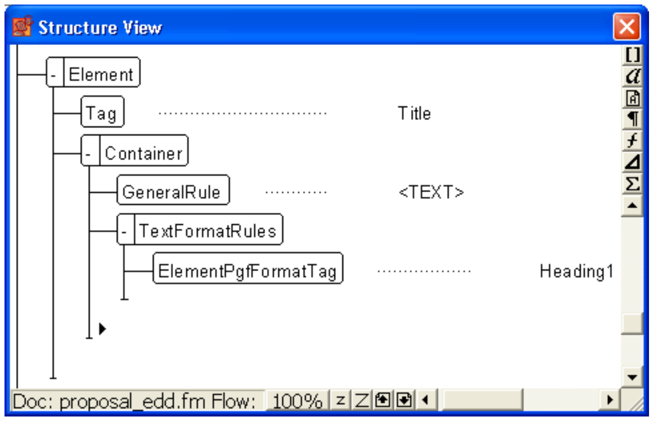
3 In proposal_EDD.fm, modify the Title element definition to use the Heading1 paragraph tag. Add the same TextFormatRules and ElementPgfFormatTag elements as you did for the Para element.

Note: *Like the paragraph tags, the information you enter is case-sensitive and space-sensitive. For example, "Heading1" is not the same as "heading1" or "Heading 1."*

Next, add a prefix rule to the Title element. Prefix rules let you specify text that should appear at the beginning of the element. Based on the Title's position, you'll specify which text should be displayed.

To add a prefix rule:

1 Position your cursor in the Title element to insert a child of Container after TextFormatRules, as shown in the following figure.

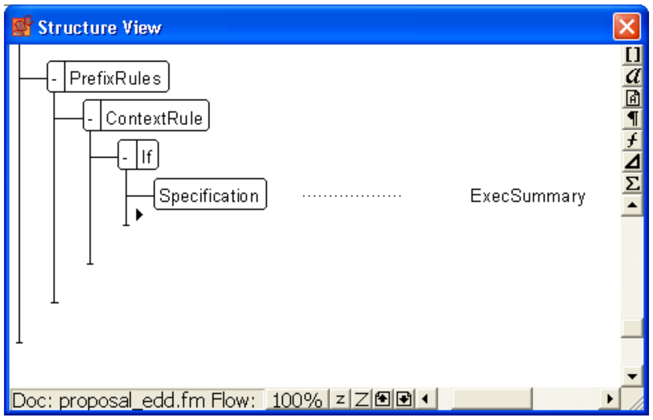


The cursor is positioned to insert a child of Container

2 Insert a PrefixRules element.

3 Insert a ContextRule element. The If and Specification elements are inserted automatically. For the Specification text, type `ExecSummary`.

4 Position your cursor underneath the Specification element (as shown in the following figure) and insert a Prefix element.

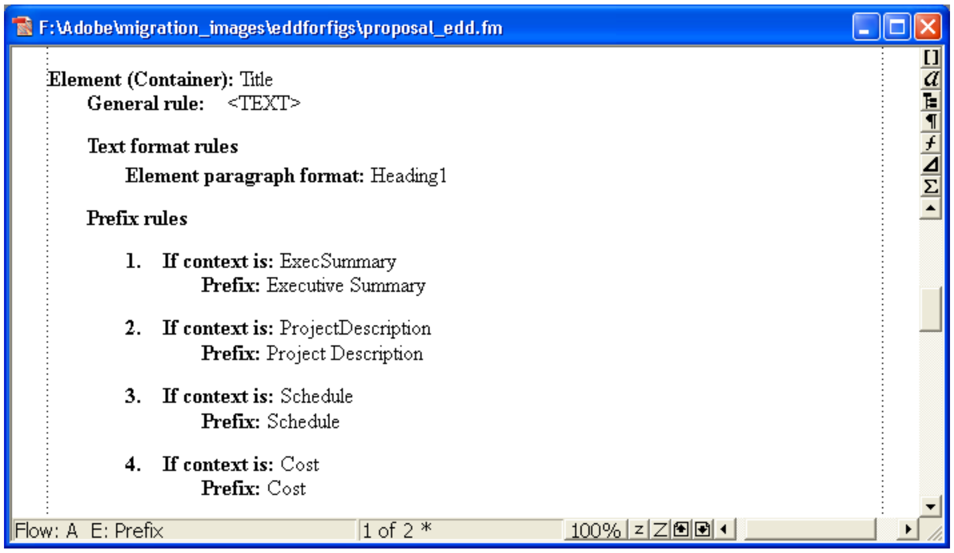


The cursor is underneath the Specification element and to the right of the If element line

5 For the Prefix element text, type:

Executive Summary

6 Repeat steps 3–5 for the other elements that need titles—ProjectDescription, Schedule, and Cost—and insert the appropriate text for each prefix. The results are shown in the following figure.



F:\Adobe\migration_images\eddforfigs\proposal_edd.fm

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

Text format rules
Element paragraph format: Heading1

Prefix rules

1. **If context is:** ExecSummary
   Prefix: Executive Summary

2. **If context is:** ProjectDescription
   Prefix: Project Description

3. **If context is:** Schedule
   Prefix: Schedule

4. **If context is:** Cost
   Prefix: Cost

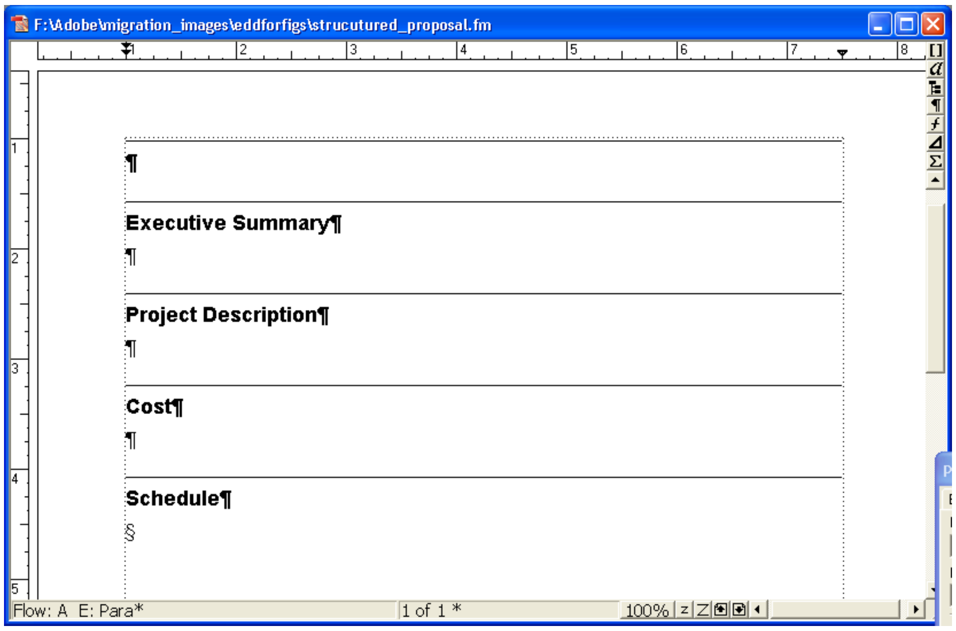
Flow: A E: Prefix 1 of 2 * 100%

Context rules for Title element

To test the results:

1 Save the EDD.

2 Import the element definitions into your structured_proposal.fm file to test the results. Each section should display the title text you've specified.


F:\Adobe\migration_images\eddforfigs\strucutured_proposal.fm

¶

Executive Summary¶

¶

Project Description¶

¶

Cost¶

¶

Schedule¶

§

Flow: A E: Para* 1 of 1 * 100%

Testing the context rules

The completed EDD is available at:

www.adobe.com/products/framemaker/proposal_edd

Building the structured application

To enable XML import and export for structured proposals, you need to set up a structured application, which lists the components of the structured authoring environment.

First, you need to create a document type definition (DTD) that matches the EDD.

To create a DTD:

1 Open the proposal_edd.fm file in structured FrameMaker.

2 Select File > Structure Tools > Save As DTD. Specify the file name (proposal.dtd) and where you want to save the file, and then click Save.

3 In the Use Structured Application dialog box, leave the default selection, <No Application>, and click Continue.

4 In the Select Type dialog box, select XML and then click OK. The DTD file is written out to the location you specified.

You also need a structured template that contains formatting and EDD information.

To create a structured template:

1 Open the structured_proposal.fm file.

2 Delete all content from the file. The structured template must be empty.

3 Save the file as proposal_template.fm.

Now, you are ready to configure the structured application.

To configure the structured application:

1 Select File > Structure Tools > Edit Application Definitions. This command opens the structapps.fm file (installed in FrameMaker's structure directory) where application definitions are stored.

Note: *Like the EDD, the application definition file is itself structured.*

2 Insert an XMLApplication element as a child of the StructuredSetup element. Type `Proposal` for the application name.

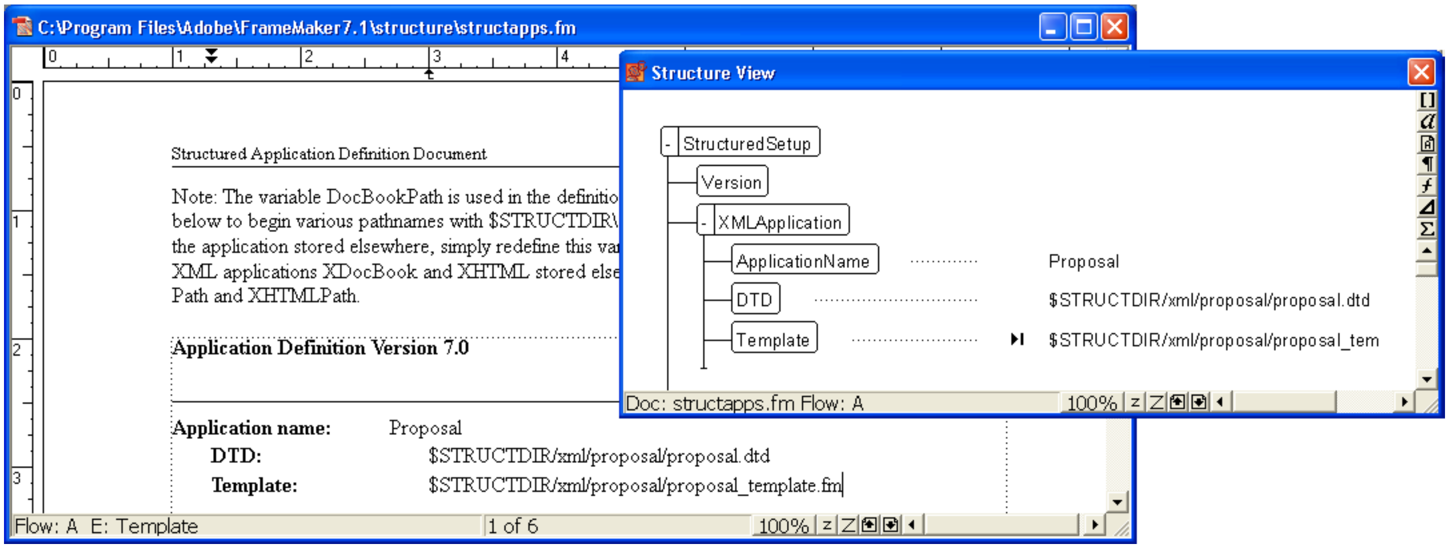
3 Insert a DTD element and specify the following as the DTD element text:

```
$STRUCTDIR/proposal/proposal.dtd
```

4 Insert a Template element and specify the following as the Template element text:

```
$STRUCTDIR/proposal/proposal_template.fm
```

The completed application definition is shown in the following figure.



C:\Program Files\Adobe\FrameMaker7.1\structure\structapps.fm

Structured Application Definition Document

Note: The variable DocBookPath is used in the definitio
below to begin various pathnames with $STRUCTDIR\
the application stored elsewhere, simply redefine this va
XML applications XDocBook and XHTML stored else
Path and XHTMLPath.

Application Definition Version 7.0

Application name: Proposal
DTD: $STRUCTDIR/xml/proposal/proposal.dtd
Template: $STRUCTDIR/xml/proposal/proposal_template.fm

Flow: A E: Template | 1 of 6 | 100%

Structure View

```
StructuredSetup
  Version
  XMLApplication
    ApplicationName ........ Proposal
    DTD ........ $STRUCTDIR/xml/proposal/proposal.dtd
    Template ........ $STRUCTDIR/xml/proposal/proposal_tem
```

Doc: structapps.fm Flow: A | 100%

Structure definitions for the Proposal application

5 Select File > Save to save the structapps.fm file.

6 Select File > Structure Tools > Read Application Definitions to update FrameMaker with the new application definition.

***Note:** To verify that the application is installed correctly, select File > Set Structured Application to display the Set Structured Application dialog box. If you set up the applications definitions correctly, Proposal appears in the drop-down list.*

Finally, you need to copy the application files into the directory where the structured application definition expects them.

To copy the application files:

1 Locate your FrameMaker installation directory using the file explorer (the default on a Windows machine is C:\Program Files\Adobe\FrameMaker 7.x).

2 Navigate to the structure\xml directory.

3 Inside the xml directory, create a proposal directory.

4 Copy the proposal.dtd and proposal_template.fm files into the proposal directory.

Testing XML round-tripping

Once you have configured the structured application, you can test XML round-tripping.

To test your implementation:

1 Create a valid structured proposal in FrameMaker.

2 Export the FrameMaker file to XML. To do so, select File > Save As and choose XML as the output format. If prompted, select the Proposal application.

3 Open the exported XML file in a text editor or an XML editor and verify that you have a valid XML file.

4 In an XML or a text editor, modify the file you just exported. You can also create a new XML file that validates against the proposal DTD. Save the new XML file.

5 In FrameMaker, open the XML file. Verify that the XML file imports correctly and that formatting is applied automatically.

Implementing structured FrameMaker

Implementation scenarios are rarely as straightforward as the proposal example outlined in the preceding section. Before you begin building an EDD and a structured application, some planning is necessary.

Content analysis

The term *content analysis* describes a process in which you review existing documents to understand how they are designed and what implicit structure they contain. Begin your analysis by making a list of the documents your organization produces. Your list might include user guides, reference guides, white papers, tutorials, training manuals, and online help. Make a list of the major components for each document type. For each component, list whether the item is required, optional, and how often it occurs in that document. The following table shows a high-level analysis for a book:

COMPONENT	REQUIRED OR OPTIONAL?	OCCURRENCE
Front matter	Required	1
Table of contents	Required	1
Preface	Optional	0 or 1
Chapter	Required	2 or more
Appendix	Optional	0 or more
Glossary	Optional	0 or 1
Index	Required	1

For major components, such as lessons (for training manuals) or chapters (for books), work your way down the document hierarchy into smaller and smaller chunks, until you reach the bottom level of the hierarchy.

COMPONENT	REQUIRED OR OPTIONAL?	OCCURRENCE
Glossary	Optional	1
Title	Required	1
GlossEntry	Required	1 or more
Term	Required	1
<TEXT>		
Definition	Required	1 or more
<TEXT>		
CrossRef	Optional	1 or more

You must do a content analysis for every document type you want to create in your structured environment. Once you have developed a content model for each document, you can look for opportunities to refine the model and reuse names. For example, an online help deliverable might consist of a collection of topics. Chapters in a book might also consist of topics. If you create a Topic element that is usable for both printed and online materials, you can minimize the number of elements in your structure definition and perhaps reuse information in both output formats.

PRINTED TOPIC	ONLINE TOPIC	COMBINED TOPIC
Topic	Topic	Topic
Title	Title	Title
Para	Para	Para
Steps	Steps	Steps
Can contain Para, List, Table, Graphic, Note	Can contain Para, List, Table	Can contain Para, List, Table, Graphic, Note
	RelatedTopics	RelatedTopics

It's unlikely that your documents are perfectly consistent. You need to decide whether to create a structure that is loose (allowing many variations) or strict (very few variations are permitted). You will have problems at both ends of the spectrum:

- A very loose structure is quite complex, and can be difficult to maintain because so many variations are permitted.
- A very strict structure may disallow element combinations that are needed.

It's very difficult to find the right balance between these two extremes. One approach is to allow variations only where they add value. For example, the content of a glossary should be fairly straightforward. You could also create reference materials (such as API documentation) that is highly structured. A standard online help topic, however, is likely to require a significant number of choices.

Note: *A similar problem occurs when creating unstructured FrameMaker templates. You want to minimize the number of tags that must be maintained while providing all of the tags that are needed to create your content.*

Once the analysis is complete, you can decide whether to use an existing structure or to build your own. Numerous XML implementations are available; DocBook and the Darwin Information Typing Architecture (DITA) are specifications used for technical documentation. Standard structures are also defined for military documents (mil-specs or mil-standards), aircraft maintenance manuals, and others. The following table describes some of the factors that influence whether you will use a standard specification or build your own structure.

USE A STANDARD	BUILD YOUR OWN
You are required to deliver content that follows the standard. For example, many U.S. military contractors are required to deliver documentation that follows a published standard.	You want to create a structure that matches your content analysis precisely.
Your content requirements closely match an existing structure. You only need to make minimal changes to the standard structure.	Your content analysis indicates that your information does not match existing structures very well.
You do not want to spend a significant amount of time building a structure, and you are willing to change the organization of your content to fit it into an existing structure.	The structure needs to match the content precisely. A longer implementation period is an acceptable cost to ensure you can build exactly what you need.
You do not have the technical expertise or resources available to build your own structure.	You have resources available, whether in-house or as consultants, who can build the structure.

Building structure files

When you complete your content analysis, you will have structure maps similar to the ones shown in the preceding section. Decide whether to use an existing standard or create your own structure.

To create structure files without using an existing standard:

1 In FrameMaker, create an EDD that matches your structure requirements.

2 Test the EDD to verify that the structure definitions are correct.

3 Add formatting to the EDD.

4 Test the EDD to verify that the formatting works correctly.

5 Create a DTD from the EDD.

You now have the two structure definition files you need—one for FrameMaker (EDD) and one for XML (DTD).

To use an existing standard:

1 Obtain a copy of the standard files.

2 Modify the provided DTD or Schema file to match your requirements.

3 Open the completed structure file in FrameMaker to create an EDD.

4 Add formatting to the EDD.

Note: *FrameMaker includes several variations of DocBook as sample structured applications. If you plan to create DocBook-based structure, consider modifying one of the provided applications.*

Once you have your structure definitions, you need to set up the structured application.

To set up the structured application:

1 Open the structapps.fm file (select File > Structure Tools > Edit Application Definitions).

2 Add a new application definition to the file with pointers to your EDD and DTD/Schema files.

After setting up the application, you'll want to fine-tune the import/export settings.

To fine-tune the import/export settings:

1 Set up a read/write rules file with the necessary mappings.

2 Set up XSL transformation files with any additional changes.

3 Add a reference in the structured application definition to the read/write rules file and the XSL transformation files.

4 Add any other required configuration settings, such as handling of conditional text and external cross-references, in the structured application definition.

Migrating unstructured files to structure

FrameMaker provides a mapping feature to help you transfer your unstructured documents into structure. Your results depend on the following factors:

- Document consistency. Documents that implement a formatting 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. A new document structure that is similar to the organization in the unstructured documents eases the conversion process.

Mapping tags to elements

The conversion process creates structured elements from FrameMaker formatting components, such as paragraph tags, character tags, markers, cross-references, and table components.

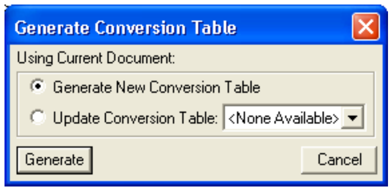
To begin the conversion process, select a document that is representative of your typical content. Ideally, this document should contain examples of all of the formatting tags that would occur in your documents. These tags should be shown in logical sequences (as they would occur in documents), so a formatting template that shows examples of each paragraph tag in alphabetical order is not a good example document.

To configure the conversion:

1 Open the example document.

2 Import element definitions from the EDD into the example document.

3 Select File > Structure Tools > Generate Conversion Table. Select Generate New Conversion Table, then click Generate.



Using Current Document:

(●) Generate New Conversion Table

( ) Update Conversion Table: <None Available>

Creating a conversion rules table

FrameMaker scans the document and creates a list of the formatting components that occur in this document. Tags that are defined in the formatting catalogs but not used in the document are *not* included in the list.

Wrap this object or objects	In this element	With this qualifier
P:Title	Title	
P:Body	Body	
P:Heading1	Heading1	
P:Code	Code	
P:Note	Note	
P:Heading2	Heading2	
P:Anchor	Anchor	
P:Caption	Caption	
P:NumberedText1	NumberedText1	
P:NumberedText	NumberedText	
P:BulletedText	BulletedText	
P:BulletedText2	BulletedText2	
P:IndentedText	IndentedText	

Excerpt of a new conversion rules table

Note: *FrameMaker assumes that the name of the formatting component will be the same as the name of the structure element.*

4 Modify the mapping rules to match the structure. For example, FrameMaker assumes that the formatting tag names match the element names, so in the preceding example, the Body paragraph (P:Body) is mapped to the Body element. To change this mapping, change the second column ("In this element") to read Para instead of Body.

5 Once you have mapped all of the formatting components, add additional entries to the table to create hierarchy. For example, if a Section element typically contains a Heading and one or more Para elements, you add a row to the table and specify how to create the Section element.

P:Title	Heading	
P:Body	Para	
Heading,Para+	Section	

6 *(version 7.2)* Add a root element mapping that specifies the top-level tag in the document, as shown here:

RE:RootElement	Proposal

7 Save the conversion rules table.

8 To test the conversion rules table, open your example document, then select File > Utilities > Structure Current Document. Select the conversion rules table document in the drop-down list, and then click Add Structure. FrameMaker creates a new, untitled, structured document.

Keep refining and testing your conversion rules until you are satisfied with the document produced.

You can add tags to the conversion rules table by typing them or by scanning additional documents.

To add tags automatically:

1 Make sure that the conversion rules table is open. Open the file that contains additional formatting components.

2 Select File > Structure Tools > Generate Conversion Table. Select Update Conversion Table and select your conversion rules document in the drop-down list.

3 Click Generate. FrameMaker scans the second sample document and adds additional formatting components to the end of the conversion rules table.

Conversion rule examples

The order in which conversion rules are listed is significant. You must go from lower-level elements to higher-level elements. For example, assume that you have the following mapping rules:

G:	Graphic
P:caption	Caption
E:Graphic,E: Caption	Figure

The rule in which Graphic and Caption are wrapped into a Figure element must occur *after* the rules in which Graphic and Caption are created.

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:

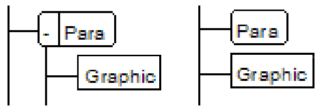
bullet	ListItem	b
step1	ListItem	st
step2+	ListItem	st
E:ListItem[b]+	UnorderedList	
E:ListItem[st]+	OrderedList	

(version 7.2) To specify the root element of a document, you use the following:

RE:RootElement	Chapter

You can only specify one root element per conversion table.

Graphics and tables are often anchored into the preceding paragraph in the unstructured document. When you structure the document, the Graphic and Table elements end up as children of the preceding Para element (shown in the following figure on the left).



The Promote command lets you create sibling elements

If you want the Graphic element to be converted as a sibling of Para (shown in the preceding figure on the right) rather than a child, use the "promote" command:

G:	Graphic(promote)

Handling conversion challenges

If the organization of your unstructured FrameMaker files matches the structure you have defined, you will be able to automate almost all of the conversion process. There are, however, several situations that make conversions more challenging. These scenarios include the following:

- Mapping from a single paragraph tag to multiple elements. For example, you might have an unstructured document that uses a Heading1 for all major sections within a chapter. In your structure, you have specified that a Chapter contains several different types of sections—Procedure, Concept, and Reference. If all of the sections have the same organization, you will probably have to review the content to determine whether a particular Heading1 should become a title in a Procedure, Concept, or Reference element. The best strategy is probably to map everything to the element that occurs must often, and then make changes manually as needed.
- Adding metadata. Your new structure probably contains attributes, such as an author attribute for the top-level Help element. If that information is not present in the unstructured file, you may have to manually insert the attribute value.
- Inconsistent document organization. If the organization of the unstructured document does not match the organization required by the EDD, you'll need to change the content to match the required structure.

Automating conversion tasks

As you develop your conversion rules table, you will discover some conversion problems that require manual cleanup. We strongly recommend that you use a scripting utility, such as FrameScript, to automate these tasks. Here are some examples (from simple to more complex) of tasks you could automate:

- Deleting extraneous empty paragraphs.
- Changing text bolded with the Bold button (or Ctrl-B) to text bolded with a character tag. You can then map the character tag to the appropriate element. (You could also use the Create and Apply Formats utility to assign new character tags for mapping. However, you must use Create and Apply Formats for all overrides; if you want to disregard some overrides, a scripting solution is more flexible.)
- Identifying and processing paragraphs that require unique treatment. In a well-designed template, converting Note paragraph tags to Note elements is easy. You can just map them in the conversion rules table. Imagine, though, that instead of using a Note paragraph tag, your content contains note information that uses the Body paragraph tag. The authors have typed in the word "Note" at the beginning of each Body paragraph that is being used as a note. (This inefficient approach is more common than you might think.) You cannot identify the Body paragraphs that are being used as notes through the conversion rules table. Instead, you can write a script that locates the problem paragraphs, removes the typed-in "Note" text, and applies a temporary Note paragraph tag. In the conversion rules table, you can then map the Note paragraph tag to the Note element.
- Assigning different elements to table row and cell elements based on the table tag. For example, you might have TableRow and TableCell for regular tables, but FieldName and FieldDescription for others. If the table paragraphs use the same paragraph tags, conversion rules cannot distinguish between the two types of table rows and cells. With scripting, you can post-process the structured document and assign specific table tags based on the parent Table element. That is, any table that uses the generic Table element gets the TableRow and TableCell elements. Any table that uses the FieldTable element gets the FieldRow element for rows, the FieldName element for the first cell in a row, and the FieldDescription element for the second cell in a row.

If you can write down a generic process for the cleanup task, you can probably automate the task with a script.

Handling formatting overrides
You can work around many formatting overrides. Before you generate the conversion rules table, create unique formats for each paragraph and character tag override by selecting File > Utilities > Create and Apply Formats. You can then map each of the new tags in the conversion rules table.

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/) for links to additional white papers, customer success stories, and more.

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

DocBook

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

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:

partners.adobe.com/public/partnerfinder/tp/show_find.do

Consulting

Adobe does not maintain an official list of FrameMaker consultants. You can, however, find them by doing a web search. Many of the training providers also provide consulting services.

FOR MORE INFORMATION

For a comprehensive overview of FrameMaker 7, please visit www.adobe.com/products/framemaker/.

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