2 Building a Business Case

Sometimes it can be difficult to convince management to accept new thinking or even the idea of saving money. This chapter discusses what you'll need to do to influence your management to implement Easy Maintenance Documentation practices.

Reasons to implement Easy Maintenance Documentation

Cost savings is the easiest way to motivate management to do something they aren't doing now or to stop doing something they are doing now.

The reasons below largely talk about "soft costs" rather than "hard costs." When you spend $50,000 to move your company to a new location, that is a hard cost. When the productivity of a group of writers decreases because of a "cost-saving" or down-scaling measure, the cost to produce documentation increases. This specific cost increase is a "soft cost" and is notoriously difficult to track.

Reason One: Save Time

In this case, implementing Easy Maintenance Documentation will save the writers time, which translates into saving money. If it takes five minutes to do something one way, and 5 seconds to do it another way, that is a potential savings of four minutes, 55 seconds each time that action is carried out. Let's compare how long it takes to manually update a block of text in five locations versus using a text inset. In this example, it takes a writer one minute to update a block of text once. Multiply that by five locations and that is five minutes. Add in the time to open each of the remaining four locations (assuming you have some way of tracking all the places that the block of text is used), perhaps three minutes to open, Find/Change, save, and close each file, that comes to 12 minutes. Total time, 17 minutes.

If the writer used a text inset, she would go to one location and update the text; that is one minute. Every place it is used, it is updated; thus time to update all five files is one minute. That's a savings of 16 minutes. At the conservative estimate of $40/hour for a writer's time, that translates to a savings of $10.67 each time the text needs to be modified.

Another example of an easy way to save time (read: the company's money) would be to use a variable instead of typing it manually. Then when the definition for the variable needs to change, the writer changes it in one location and it is automatically updated everywhere it is used. The writer doesn't have to Find/Change through all 400 pages of the document. Say that the writer typed the name of the product in 100 places in a document of 10 FrameMaker files in a book. To Find/Change through all the files in the book (assuming you're using FrameMaker 6/7) might take three minutes per file. That's 30 minutes (it is much longer if you're using v5.x). By using a variable, the writer changes the definition in one place, which might take a minute, and then imports the definition to all the files in the book, which might take another three minutes. Total time, four minutes. That's a savings of 26 minutes, or $17.33.

Reason Two: Adherence to Enterprise Styles

By doing everything outlined in this book, you will ensure uniformity of style and adherence to a style sheet.

This can be of great importance; one company spent $250,000 to have their templates created. Understandably enough, they wanted all the documents to look the same.

Even if your company hasn't spent $250K on templates, if you have gone to the trouble and expense to create templates, you want to use those styles. Another aspect of adherence to templates is the conversion of FrameMaker documents to Structured FrameMaker; by using existing formats in the template, paratags, chartags, and tabletags can be easily converted to elements, and then saved to SGML. When converting FrameMaker/Structured FrameMaker documents to HTML using WebWorks Publisher, if users strictly adhere to the styles in FrameMaker, the conversion to HTML (using the templates created in WebWorks Publisher) will be nearly effortless. If not, someone will have to go back into the FrameMaker/Structured FrameMaker documents and remove overrides and correctly apply formats.

Reason Three: Improved Morale

To troubleshoot a series of templates (or to create a series of templates), first sit down with the users and listen to their problems. One of the main complaints you may hear is that they don't like the templates. After a bit of digging, you'll get past the generic objections and uncover the real reasons, which usually turn out to be one of the following:

- The users don't like the names of the formats.
- They don't know how to use the existing formats, so they create their own.
- Someone else designed the formats and they didn't have a voice in the creation.

The Second Tenet of Easy Maintenance Documentation Uniformity

The Second Tenet is: Every document within in a group of related documents that uses the same set of formats must have the exact same definitions for all formats.

This means that all variables, paratags, x-ref formats, condtags, master pages and so on, must have the same definitions throughout the book. FrameMaker's ability to create format overrides, such as hardcoded text formats or local variations on a chartag, paratag, or tabletag, is diametrically opposed to the Second Tenet.

This chapter discusses the tools that enable you to ensure uniformity of formats across your entire enterprise or just across your own document set.

As mentioned in the First Tenet, having a single source allows for global management of formats. In this chapter, you'll see how uniformity of styles also means global management of styles.

The Second Tenet explains the following:

- How to use a Local Template to ensure uniformity across documents
- How to control formats across many documents
- The process for creating a template
- How to create the various formats that comprise a template

Master Template and Local Template

Imagine a tool that gives you the power to control the overall appearance of all the documents across your entire enterprise, at discrete levels and within groups of documents.

This tool allows you to define how all the product names, logos, addresses, and phone numbers, for your entire company can be spelled and formatted.

The tool is tightly integrated to work with every FrameMaker product on every platform.

Imagine that you that you don't have to spend any money to buy this powerful tool; it is part of FrameMaker.

Where is this tool? It is called the Master Template. It doesn't exist in any FrameMaker menu and it isn't documented in any FrameMaker manual or any third-party FrameMaker book, except this one.

The Master Template is largely conceptual rather than an actual tool, so the use of the Master Template requires conceptual training to understand it and discipline to use it.

At the highest level within a company, you use a Master Template to control formats that are used in every document produced by the company. Then as the focus becomes more specific, you might use Site Templates, Workgroup Templates. When you get to the finest level of granularity in this concept, at the individual document level, you use Local Templates to create and modify formats that are specific to the document.

Differentiating Between Templates and Master Template/Local Template

FrameMaker uses some terms differently, so this section defines the differences between them.

Template

As used by FrameMaker and *Big Docs Made Easy*, a template is a series of formats stored within a FrameMaker document that can be used to create a new FrameMaker document. Usually, a template is saved to the Templates directory. The Templates directory is special: any document stored therein can be used to create blank documents containing all the pre-defined formats, without affecting the original document.

Master Template

In this book, the term Master Template is used to describe a place in which all the common definitions used within a closed group of documents are stored, maintained, and referred to. There can be as many levels of Master Templates within a company, site, or workgroup as necessary. Here is an example: WorldCall uses four levels of Master Templates: Global, Business Unit, Site, Workgroup. At the document level, they have local-templates.

Local Template

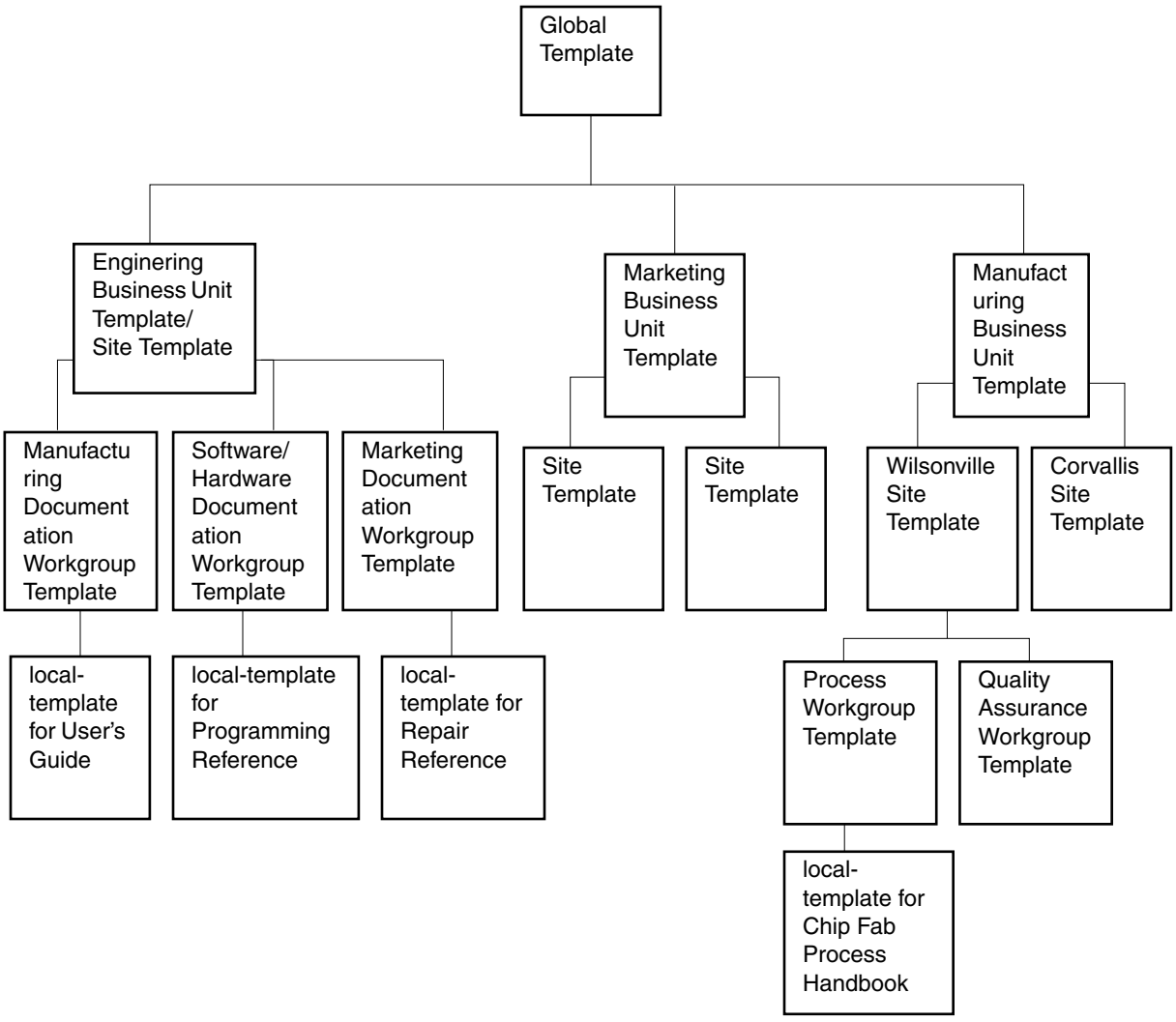
The Local Template is a repository for all formatting information pertinent to a particular book. This is similar to a Master Template, but at the lowest level in the hierarchy. For example, the Master Template might define master page layouts for all documents, such as Right, Left, Cover, FrontMatterRight, FrontMatterLeft, FrontMatterCover, IXRight, IXLeft, and IXCover. Each master page also contains a placeholder variable for the manual name, part number, and revision level. The Local Template allows you to define a value for each of the placeholders for the book.

Master Templates and Local Templates are the embodiment of the Second Tenet; using them enables you to ensure that all formats used in all of the documents in a book will have the exact same definition, and hence, a uniform appearance for every document in the book.

When you use and enforce uniformity, you are implementing the Second Tenet; you won't have to worry about overrides created by users (which they then forgot about and someone else inherits). When you create robust formats, use them uniformly, and maintain them from one place rather than from several places, you are implementing the the First Tenet. Another way to look it is, if the the First Tenet is the heart of Easy Maintenance Documentation, then the Second Tenet is its soul.

Example of Master Template / Local Template Implementation

WorldCall (**WC**) has graphic standards and trademarks that must be used in specific ways to protect their trademarks. They created a Global Template that is used as a basic building block for every subordinate Business Unit Template. It contains logos, typefaces, addresses and phone numbers that must be used in specific ways. When they are included in the Templates at the next level down, these are automatically controlled. The Global Template is controlled from the world-wide headquarters and is rarely changed. However, **WC** has business units on three continents, and sometimes the language used is different. For each business unit, a different Master Template was created that contains some localized addresses and phone numbers, and some solutions to problems that only occur in FrameMaker with certain languages, such as Chinese, Japanese, or Korean.



Within each business unit of **WC**, there are sites scattered hundreds of miles apart from each other or have completely different product lines. It would be a waste of resources to carry formats in the Global Template for one business unit that are not used by other business units. Therefore, the reason for the hierarchy of templates is to manage the formats that are common to all documents that use those formats.

Controlling Formats Across an Enterprise

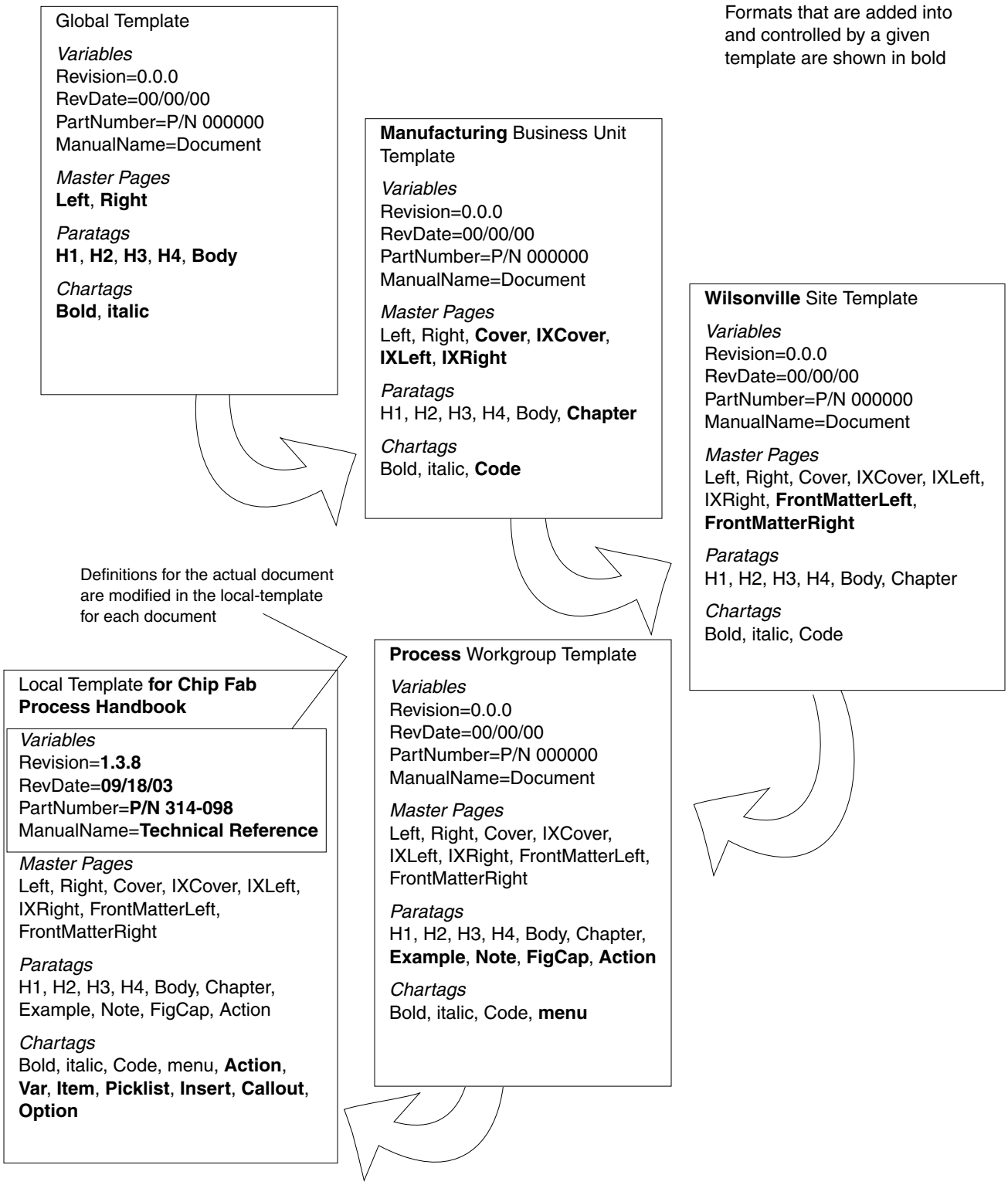
Within each site, there may be as many as ten different Workgroups. Each workgroup has different requirements for presenting information: some workgroups provide training documentation. Other workgroups provide end-user documentation. Each workgroup decides how it will document its particular type of work and what is required in their template.

Finally, each document requires different standards, different variables and so on. To carry document-specific variables at any other level would be a tremendous waste of resources. However, that is not to say that at some time, a writer within a workgroup couldn't petition the administration to include in the parent template a solution, variable, or some other piece of often-used text.

This concept of one template controlling many can be scaled up or down as enterprise needs dictate.

Each level of the template adds in only those formats that will be managed for subsequent layers of templates.

Here is how one branch of the Templates tree concatenate formats from the Master Template down through the various other levels of intermediate templates to the Local Template for a specific document:



Controlling Formats within a Book

Let's suppose that you are doing all the things that were recommended in the first section; you are happily creating your variables and using them throughout your book. Then a subject matter expert (SME) calls you to say that the names of three of the fields have changed. No problem, you think. You have defined them as variables. You open one of the chapters of the book, open the variables dialog box, change the definition and you're done. But what about the other chapters where you need to change the definition? Do you open all the chapters in the book and change the definition in each one?

There is an easier way: change the variable definition in one place and import that definition to all the other chapters. Two easy steps, rather than many repetitive steps. Yes, you could do it by simply importing the changed chapter to all the other chapters of the books, but you may find it better to obey the First Tenet and create a central location that contains all the Source, change it there and then import those changes to everywhere else. Until FrameMaker comes up with a better scheme of updating global definitions, this works well. But it presupposes a few things: that you have defined your master pages to have different names when they have a different appearance in other documents. You may inherit documents where the master pages for the TOC had the same names as the document pages but were laid out differently. When you import the formats from a document into a TOC, the TOC master page definitions suddenly look just like the normal document pages. So, when you import formats, make certain that you have created your master pages to have unique names.

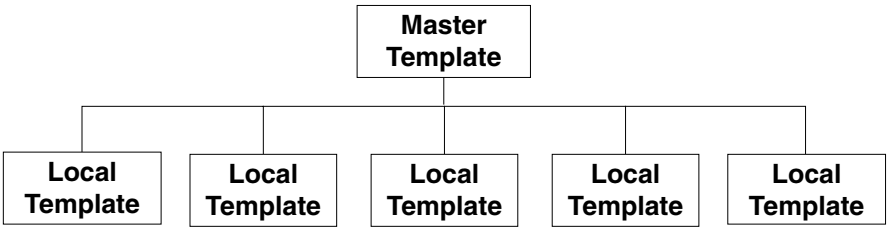
Note: For those among you who are saying, "Just uncheck the Page Layout option in the Import formats dialog box and import the other formats"; if you ever forget to do that, your page definitions are likely to be overwritten. It's easier to have unique names for unique formats, than forget and have to recreate them.

Having a Master Template also allows you to choose if a paragraph tag or character tag that you create for one document can be re-used in other documents. You may create paragraph tags for one document that aren't needed anywhere else in the document; in this case, leave them there and don't import them into the Master Template. However as soon as you use them in more than one document, import the definition into the Master Template and import the definition into the rest of the book.

Note: Make CERTAIN that you have unique names for every format or style; if you don't, as soon as you import definitions to all documents they will be overwritten. This can backfire when you have modified a paragraph tag in one document; this is a format override as well as a violation of the Second Tenet.

Levels of Granularity for Master Template/Local Template

Your organization may be large enough to use a Global Template > Business Unit Template > Site Template > Workgroup Template > Local Template hierarchy, but you may not be able to implement a Master Template hierarchy above or even within your own workgroup. You can still implement a Master Template/Local Template setup just for your documents. A simpler version would be to have only one Master Template that controls a large group of Local Templates, as shown below:



You could also organize your Master Template/Local Template by document types rather than by workgroups.

Implementing Master Templates

There are two ways to implement Master Templates: Open Access and Controlled Access.

Open Access

With open access, the template administrator has read/write access to the Master Template. The users have read-only access to the Master Template and full read/write access to their Local Templates. By allowing users to have access to the Master Template and to their Local Templates, users can open and import the formats contained in the Master Template into their Local Templates. The users can then also modify the Local Templates for their documents to add paratags, chartags, x-ref formats, and variables.

All formats, master pages, and reference flows are contained in the Master Template. When definitions are changed in the Master Template, the users import the definitions into their Local Template(s) and then clean up their Local Template(s). The Local Template contains locally changed variables, such as document names, system names, manual numbers, revision numbers, revision date, and so on. In this way, the Local Template is under the control of the user rather than the template administrator and allows some flexibility on the part of the user.

Advantages

- The maintenance of the Local Templates is the user's responsibility. If the user knows how to use FrameMaker, this can reduce the burden on the template administrator.
- Users can be in control of their own documents, have discretion in creating new formats, and not be forced to "toe the line" in every regard.
- Template administrators are peers, not petty tyrants of style.

Disadvantages

- Users may add local definitions that may conflict with other layout requirements, or content repurposing templates (such as XML/HTML).
- The maintenance of the Local Templates is the user's responsibilities. If the user is not familiar with FrameMaker, he or she could mismanage the Local Template and get bogged down in administring the Local Template.

Controlled Access

The other method is controlled access to Master Templates and Local Templates. The template administrator is responsible for updating the definitions contained in the Master Template, importing those definitions into the Local Templates of all documents, and then modifying the definitions for local formats and variables in each Local Template. Only the template administrator has read/write permissions to the Master Template and to each documents' Local Template. The Local Templates can be read by the users but cannot be modified. This ensures that all changes to the documents are known and agreed to by the template administrator.

In this method, the user imports the definitions from the Local Template into the documents when the Master Template changes. It is the template administrator's responsibility to notify users of changes in the Master Template.

A whimsical way to think about controlled access: it is training wheels for structured documentation using XML or SGML. When you use structured documents, the DTD (Document Type Definition) or EDD (Element Definition Document) and stylesheets are used by the entire group and must be administered centrally.

Advantages

- Absolute control of all formats, and ultimately, the appearance of all documents.

Disadvantages

- The users may chafe at the inflexibility in this situation
- The template administrator position becomes a full-time job

Templates Directory

Once the Master Template is finished, create a set of blank documents (which are then stored in the Templates directory [`$FMHOME/fminit/usenglish/Templates` or `..\Program Files\Adobe\FrameMaker\Templates`]) that have only the approved formats in them. Users then create new documents by opening copies of the Templates.

Blank Templates Directory This directory contains read-only versions of every type of FM File contained in the FrameMaker book for the document. That means there is a chapter template, a TOC template, an IX template, and a template for any other specialized parts of a book.

You will create this directory in Exercise 69 on page 264.

Standardizing Nomenclature

The Second Tenet contains the following corollary: standardize the nomenclature used in a project.

Assigning one standard name for a thing can really help your project. When all the writers and managers involved in a project agree on the terms used to describe the parts of a project everyone works together better. The reason is simple; when everyone uses the same term to describe something everyone knows what is being described! It is extremely hard to work with people who use different terms than the rest of the group.

For more information, see *Agreeing on Nomenclature* on page 419.

Document Analysis

When you design your document, probably you consider how it will look first and then how easy it will be to maintain. You might find it faster to answer a few questions before you design your templates to make maintaining your document a lot easier.

Begin the process by asking general questions.

- Are you writing a user's manual, a technical journal, a technical reference manual, an illustrated parts catalog, a field service/repair/installation manual, an operation, administration, and maintenance manual, a process manual, or a product specification document?

Let's assume the answer from the previous question is a user's manual; now you can start to focus on the needs of the type of document.

- Do you need complex page numbers (i.e., page 5-24) or will consecutive page numbering (i.e., page 125) be accepted by readers?
- Will your document be printed or viewed on-line? If printed, you'll want double sided; if viewed on-line, you'll want single-sided.
- Do you want the first page to look different from the rest of the pages?
- Do you want complex section numbers (i.e., 5.4.4 Optimizing the Link Layer)?
- Do you want the chapter title/appendix title text or chapter number/appendix letter to appear as a Running/Header Footer?
- Do you want the section title number or text to appear as a Running/Header Footer?
- Do you want the front matter of your book (i.e., preface, TOC, LOP, LOT, LOF, etc.) or the index to look different from the main part of your book?
- Will your document re-use text that is already written?

- Do you have extensive procedures, extensive description and theory, or a mixture of both?
- Do you have company logos, company specific nomenclature, different names for the same product directed at different markets?
- Do you have to reference specific text or programming code that cannot be altered only read?

Choosing a Template Style

There are too many ways to lay out a template to reasonably list them all. However, there are discrete template style categories, which can be listed:

1 Side Heads on, with In Column and Across All Columns and Side Heads (AAC+SH) headings

2 No Side Heads, with Indents to emulate Side Heads.

3 With Chapter-level TOCs (CTOC)

4 Without CTOCs

5 Complex Page Numbering (compound Chapter number and Page number)

a with compound autonumbering for Tables, Figures, Exercises, Examples, and Headings

b with consecutive autonumbering for Tables, Figures, Exercises, and Examples.

c no autonumbering for Tables, Figures, Exercises, Examples, or Headings

6 Simplex Page Numbering

a with consecutive autonumbering for Tables, Figures, Exercises, Examples etc., and Headings

b no autonumbering for Tables, Figures, Procedures, Examples, etc., and Headings

Use Side Heads or In-Column with Indents?

One of the main points of bifurcation is whether to use Side Heads or to use indents to emulate Side Heads. The reason that this is a big decision is that if you use Side Heads, the side head space can be turned on or off globally: it is a lot harder to change the indents of many paratags that emulate Side Heads. Also if you want to use the same paratags in table cells as well as in main body pages, you have to skip using emulated side heads because they will be indented in the table cells, which looks a bit odd, not to mention uses a lot of space in your table cells.

Turning Off Side Heads for Specific Paratags

If you use side heads, you will want some paratags to go across side heads. Unfortunately, you can't turn off side heads for individual text frames that use the same flow tag. (If text frames don't have a flow tag, you can turn off side heads for each text frame, but that is another issue.)

The only way to control this is to configure paratags to go Across All Columns and Side Heads. For more information, see *Using Multiple Columns or Multiple Text Frames?* on page 156.

In the following exercises, you will build a local-template for the most complex setup: complex page numbering with CTOCs, compound autonumbering for tables, figures, exercises, and headings with side heads.

Chapter-Level Tables of Contents (CTOCs)

A chapter-level TOC is a table of contents that has greater detail than the TOC that is created for an entire book. A book-level TOC may have only first- and second-level headings in it. A CTOC may include first-, second-, and third-level headings, tables, figures, and procedures. This much detail would be overwhelming in a book-level TOC.

Complex Page Numbers

A complete implementation of complex page numbering actually has three parts:

1. Compound autonumbering paratags so that the Chapter-level counter is correctly carried through in subordinate paratags, such as Figures, Tables, and Exercises/Examples/Procedures, etc. This is much easier to accomplish in FrameMaker 6/7 by using the <$chapnum> building block.
2. The <$chapnum> and <$currentpagenum> building blocks that are in headers or footers on the master page.
3. If you use x-refs with page numbers, they also must include the <$chapnum> and <$curpagenum> building blocks.

We'll talk about these three issues in greater detail in *Master Page Layouts* on page 141.

Simplex Page Numbering

The issue of template design becomes easier when you don't use compound autonumbering for figures, tables, exercises/examples/procedures, etc.

Note: Some user manual styles have eliminated figure numbers and table numbers; this eliminates the need to refer back to a figure several pages back because the figure or table that is relevant to the text is kept near it, such as within the same two-page spread.

Roadmap for Complex Template Creation

As we mentioned before, the Local Template, Master Template, and Style Guide that you will create by completing the exercises in this book incorporate the most difficult of formats of all the discrete categories:

- AAC+SH (Across All Columns and Side Heads) with Indents to emulate Side Heads, along with Side Heads
- CTOCs
- Complex Page Numbers
- Compound autonumbering for Tables, Figures, Exercises, Examples, and Headings

In the sections that follow, you will be given step-by-step exercises for designing a Local Template, a Master Template, and a Style Guide. Here is the roadmap for this phase:

1. Perform a document analysis (see *Document Analysis* on page 91)
2. Choose a Template design (see *Choosing a Template Style* on page 92)
3. Create paratags, chartags, x-refs, condtags, variables (see *Creating a Local Template* on page 97)
4. Understand and create the components of a master page (see *Master Page Layouts* on page 141)
   a. Set up complex page numbers (see *Complex Page Numbering* on page 146)
   b. Using Running H/Fs to extract text (see *Using Running H/Fs to create Complex Page Numbers* on page 147, *Running Header/Footers in Headers and Footers* on page 145, and *Efficient Use of Running H/Fs* on page 162)
   c. Create any other formats to fulfill your style requirements.
5. Create x-ref formats to include the <$chapnum> building block (see *Multiple Autonumber Counters Within a Series* on page 129, *Creating X-Refs to Split Paratags in One Format* on page 191, and *Using X-Refs with Complex Page Numbers* on page 181)
6. Generate all generated flies (TOC, IX, etc.)
   a. Format paratags
   b. Format master pages and reference pages
   c. Copy master pages and reference pages into Master Template
   d. Copy master pages and reference pages into Local Template (see *Creating & Maintaining Reference Flows* on page 213)

7. Create Chapter-level TOCs (see *Chapter-Level Tables of Contents* on page 194)

8. Expand the book and address general book issues (see *Book Building Tips* on page 541)

   a. Breaking up a long chapter into several documents (see *Interesting Uses for Hidden Anchored Frames* on page 377)

   b. Using Hidden Anchored Frames to carry autonumber formats (see *Interesting Uses for Hidden Anchored Frames* on page 377)

   c. Creating complex Index Entries (*Marker Tips* on page 505)

   d. Adding a Title to a TOC and an IX (see *Generated Files Issues* on page 324)

9. Build a Master Template (see *Building the Master Template* on page 251)

10. Build the Style Guide that describes how to use all the formats in the template

11. Create templates for the various parts of a book from the local-template

    Before you can create the paratags that contain the autonumbering formats, it helps to understand what can be automatically extracted and formatted on the master pages.

16 Graphics Tips and Tricks

Objectives

In this chapter, you will learn how to:

- Create automatic banner headings
- Set x-refs in graphics
- Recognize and solve graphics problems
- Create “boxed” callouts

Automatic Banner Headings

The main reason to create automatic banner headings is to avoid repositioning manually drawn and filled rectangles when text is added or deleted or the pagination changes.

PageMaker allows the format trick of a thick line above or below a paragraph to sit behind the text, so it is a simple matter to create a banner heading.

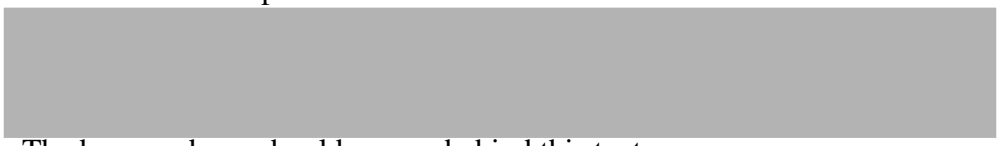
A posting to the FrameUsers website (http://www.frameusers.com) emulated a similar solution in FrameMaker. It required two paragraphs; one to hold a Reference Frame and one to hold the text with NEGATIVE space above so that the text is positioned over the banner.

There are two problems with this trick:

1. The text disappears every time you try to edit it; to make it reappear, you have to refresh the screen ([Windows only] **Window > Refresh**; or `Control-l` [lowercase L]). For an example of this, see `../bdme/ch16/banner.fm`.
2. The banner is only as wide as the graphic stored in the Reference Frame. The downside to this is if you want to have variable widths of banners for your headings, you’ll have to have many different paragraph tags and Reference Frames.

Another solution requires you to create a MIF version of a document and set parameters for `<FDY %>` for the paratag. While this is definitely the terrain of power users, it is not easily maintained. If you change any of the parameters for the paratag (using the Paragraph Designer), the customized settings for `<FDY %>` are lost and you must start again. And you’ll have to set the values for each MIF file that uses this solution, which is a potential maintenance problem.

The ostensible solution of using an anchored frame to hold a filled rectangle doesn't work either because the anchored frame (regardless of the anchoring position or type) and its contents sit above the text, rather than behind it, as shown in the example below.



The banner above should appear behind this text.

The best way to create reversed headings that automatically move with the text is to create a single-cell table.

Exercise 133 Using Tables to Create Banner Headings

1. Open `../bdme/ch16/local-template.fm` and find the Scratch Work Area section.
2. Move your insertion point to the end of this section, and create a blank paragraph (or use an existing blank paragraph).
3. Insert a table with the following settings:

Property	Value
Format	Blank
Columns	1
Body Rows	1
Heading Rows	0
Footing Rows	0

4. With your insertion point in the table, open the Table Designer.
5. In the Table Tag: box, delete the contents and type: BannerHeading
6. Click the **Commands** pop-up menu and select New Format.
7. In the New Format dialog box, confirm that both Store in Catalog and Apply to Selection are selected. Click **Create**.

8 Modify the properties of BannerHeading using the settings in the table below:

Property Sheet	Property	TableTag
		BannerHeading
Basic	Default Cell Margins	
	Top	5
	Left	5
	Bottom	3
	Right	5
Ruling	All Rulings	None
Shading	All Heading, Footing, Body Shading	100% Black

9 With your insertion point in the table, create a new paratag based on CellBody using the settings shown in the table below:

a. Open the Paragraph Designer. In the Paragraph Tag box, delete CellBody and type: BannerHeading

b. Click the **Commands** pop-up menu and select New Format.

c. In the New Format dialog box, confirm that both Store in Catalog and Apply to Selection are selected. Click **Create**.

d. Modify the properties for BannerHeading using the settings shown in the table below:

Property Sheet	Property	Paratag
		BannerHeading
Default Font	Family	Helvetica (or Arial)
	Size	14
	Angle	Regular
	Weight	Bold
	Color	White
Basic	Indents	0
	Alignment	Left
	Space Above/Below	0
	Line Spacing	17

Explanation: Once a printed document leaves your hands, for all intents and purposes, it is beyond your control. It will spend its life on various copier beds and be copied and recopied and recopied and recopied…with copies of copies of copies being distributed. It may be faxed, sometimes through several generations. If you use reversed text, that is light text on dark background, after a few generations of reproduction, the text will start to get "drop-ins," eventually rendering the reversed text unreadable. You may find that a sans-serif font (such as Helvetica) Bold, 14 points or larger, Regular (neither oblique nor italic), will stand up better to copying, faxing etc., and will be more readable in the original document.

10 With your insertion point in the BannerHeading table, type: `The easy way to make banner headings`

The text wraps and forces the cell to grow vertically

11 Resize the table so that the text doesn't wrap (unless, of course, that is the effect that you want) using one of the following methods:

- **Table > Resize Columns**, select To Width of Selected Cells' Contents, and click **Resize**.
- Press `!tz` to open Resize Columns dialog box, select To Width of Selected Cells' Contents, and click **Resize**.
- Press `!tw` to resize column(s) so no paragraphs in selected cell(s) wrap

Below is an example.

The easy way to make banner headings§

Note: This is most effective with multi-column layouts where the heading isn't very wide.
You can also experiment with Default Cell Margins to create different appearances for your banner headings.

12 To ensure that every time you create a BannerHeading table, the cells are formatted with the BannerHeading paratag, update the table definition.

a. With your insertion point still in the table, open the Table Designer.

b. Click **Update All**.

13 **Save** your work.

X-Refs in Graphics

Imagine that you have a graphic that refers to a list of parts in another place in the document. The graphic has been generated outside of FrameMaker, but you want to manage the callout text within FrameMaker.

1. Create the graphic (or have the graphic created) so that the places for the callouts are blank. You will put those in when you bring the graphic into FrameMaker.
2. Import the graphic by reference into your document. FrameMaker automatically inserts the graphic in a centered anchored frame.
3. Using the Text Frame tool, draw text frames in the anchored frame in the appropriate places where you want the callouts.
4. Put your insertion point in the first text frame and insert an x-ref to the appropriate paragraph in the list of parts.
5. Continue inserting the remaining x-refs using Serial Insertion of X-Refs. (For more information, see *Serial Insertion of X-Refs* on page 461.)

By doing this, you manage the complete document in FrameMaker and the graphics are brought in with as little modification as possible. This allows you to re-use graphics; if the graphic had specific references, you would have to change the references in the source graphic, which would be a waste of time since FrameMaker handles that sort of thing easily.

You may have created documents with callouts by manually typing Text Lines. The problem with Text Lines is that the text cannot be x-ref'd, formatted with a paratag, cannot contain a marker, and cannot be included in a generated list. By using text frames, you have text that can be tagged with a flow tag, formatted with a paratag (and then globally managed), x-ref'd to a table and included in a generated list.

A good example of how to use this is a parts diagram.

Exercise 134 Parts Diagram with X-Refs to a Parts Table

Imagine that you have a parts diagram that refers to a parts list, such as an ATA (Air Transport Association) IPC (Illustrated Parts Catalog) or a mechanical drawing and a bill of materials.

Rather than using a drawing with a parts list in it, use just the graphic and create x-refs to a separate parts table that you can create and maintain in FrameMaker.

Marching Bleed Tabs

Marching bleed tabs are graphics (usually solid black boxes) at the edge of the page farthest from the binding edge. The effect is greatly enhanced when you trim 0.20" to 0.25" off the edge of the document after it is printed and bound; the marks show up on the edge of the page and give a visual guide to the various chapters.

Over the years, many solutions have been proffered for this "problem." Really, it isn't a problem because if you really need tabs, have them inserted prior to binding. However, for small runs, having custom-made, die-cut tabs created and then inserted can increase the per manual cost by as much as $10!

Here is a low-cost solution to the Marching Bleed Tabs "problem."

Exercise 141 Creating the Paratags

This setup assumes that you want to have no more than ten bleed tabs (that is each tab is 0.875" wide, there is a space of 0.125" between each tab, and one-half inch of space at the top and bottom).

Note: This exercise requires the use of the ZapfDingbats font. If you do not have this font, this exercise will not work.

1. Create a blank portrait document. **Save As** `bleed-tabs.fm` to `../bdme/ch18`.

2. Create ten paratags `Chap1` through `Chap10`. Use Body as a starting point and modify only the Autonumber properties.
   Hint: Create the first paratag `Chap1` from Body. Then use `Chap1` as a starting to create `Chap2` through `Chap10` and you won't have to specify the Autonumber Format for each paratag.

   a. Define autonumber `C:Chapter\ <$chapnum>`

3. Go to the `Right` master page.

4. Create a background text frame of any size.

5. Rotate the text frame 90° clockwise (**Graphics > Rotate** or `!gt`).

6 Using Object Properties resize and position it using the following settings:

Property	Value
Width	10.729
Height	0.75
Offset from Top	0.083
Offset from Left	7.75

7 With your insertion point in the text frame, create a new paratag named BleedTab. Make sure to apply it to selection, but do not store it in the catalog. Use the following settings:

	Property	BleedTab
Default Font	Family	ZapfDingbats[a]
	Size	90
	Angle, Weight	Regular
	Color	Black
Basic	Alignment	Left
	Tab Stops	0.875 Center (repeating every 1 inch)

a. You MUST use ZapfDingbats or this will not work.

Note: You set the first Tab stop at 0.875" repeating every 1" for nine more tabs.

8 In the newly inserted text frame, insert the Running H/F 3 variable. Modify the definition for Running H/F 3 to be as follows:

```
\t<$paratext[Chap1]>\t<$paratext[Chap2]>\t<$paratext[Chap3]
>\t<$paratext[Chap4]>\t<$paratext[Chap5]>\t<$paratext[Chap6
]>\t<$paratext[Chap7]>\t<$paratext[Chap8]>\t<$paratext[Chap
9]><$paratext[Chap10]>
```

9 Copy the text frame and paste it.
(If you did NOT get the Add Text Frame dialog box, delete the text frame and try again).

10 In the Add Text Frame dialog box Select Background Text Frame and click **Add**.

11 Using Object Properties resize and position it using the following settings:

Property	Value
Width	10.729
Height	0.75
Offset from Top	0.083
Offset from Left	7.445

12 With your insertion point in the text frame you just created, create the BleedTabNumbers paratag. Make sure to apply it to selection, but do not store it in the catalog. Use the following settings:

	Property	BleedTabNumbers
Default Font	Family	Helvetica
	Size	48
	Angle	Regular
	Weight	Bold
	Color	White
Basic	Alignment	Left
	Tab Stops	0.875 Center (repeating every 1 inch)

Note: The tab stops should already be set because you are basing BleedTabNumbers on BleedTab.

13 In this text frame, click the Running H/F 3 variable and replace it with Running H/F 4. In the Variable dialog box, change Running H/F 4 to the following definition:

```
\t<$paranumonly[Chap1]>\t<$paranumonly[Chap2]>\t<$paranumon
ly[Chap3]>\t<$paranumonly[Chap4]>\t<$paranumonly[Chap5]>\t<
$paranumonly[Chap6]>\t<$paranumonly[Chap7]>\t<$paranumonly[
Chap8]>\t<$paranumonly[Chap9]><$paranumonly[Chap10]>
```

14 Create a new master page. Name it BleedTab and copy the layout from the Right master page.

15 Go to the body pages and apply the BleedTab master page to the current page.

16 With your insertion point at the beginning of the document, apply the Chap1 paratag and type: `n`

17 Refresh the screen by pressing `Control-l` (lowercase L) (or Windows only: from the Windows menu, select **Refresh**).

The "`n`" you typed is extracted by the Running H/F 3 and creates a black box *behind* the white number. The white number is created by BleedTabNumbers which is extracted by the Running H/F 4, as shown in the sample below.

18 If your document does not look like the sample shown above, go back and repeat the steps until you get the above result.

You must be able to display the chapter number but the text "n" must also be there for the Running H/F 3 to extract. How can the text be part of the document without actually seeing it?

You must put the Chap1 (or Chap2, Chap3, …, Chap10) paragraph in a Hidden Anchored Frame. But then what about the number: you want to see that in the document. How can you display the chapter number without the "`n`"?

You must create a generic paratag that will simply display the value for the Chapter Number variable. You already have this paratag, in `sg.local-template.fm`; it is called ChapterNumber.

Some tweaking is still required to make this work in your templates: if you decide to use this solution, you may want to have the bleed tabs also appear on the title pages. If so, simply copy the two text frames you created on the BleedTab master page and paste them on any other right-hand master page.

You'll probably want the Right master page to use this layout so you don't have to manually apply the BleedTab master page to every body page.

Also, you'll need to change the Running H/F usage a bit, since this solution uses two of them, and overwrites the original definition for Running H/F 3, which was used to inset the non-breaking hyphen in the complex page numbers. You may choose to have fewer Running H/Fs in your headers or eliminate complex page numbers, which would free up a Running H/F.

Use Chap1 for chapters 1, 9, 17, etc. Use Chap2 for chapter 2, 10, 18, etc. and so on. Use Chap8 for chapter 8, 16, 24, etc.

Since each chapter uses a different paratag, the Running H/F will only extract one number from a chapter.

The only limitation is that this has a maximum of two digits, up to 99 chapters. Beyond that, this solution will not display the numbers correctly.

Other Solutions

Unfortunately, with this solution, you can't display more than the number. Inserting several black boxes will not work because the boxes will not sit flush up against each other; thus text dropped out in the spaces between the boxes.

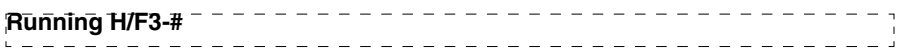
To have text displayed in your bleed tabs, create 10 different right-hand master pages, each with a Running H/F that would extract the text of a Header/Footer $1 or Header/Footer $2 Marker. The only downside to this solution is that in FrameMaker 6 you have to remember to apply the correct master page to the appropriate body page for the appropriate chapter. In FrameMaker 7, you can assign a master page to a particular paratag.

1 **Save** and **close** `bleed-tabs.fm`.

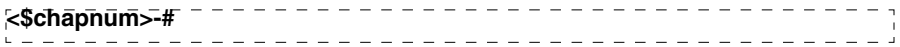
Running H/F Tips

Using Header/Footer Markers and <$marker1> and <$marker2> Building Blocks

In FrameMaker 5x, to create complex page numbers, you had to extract the autonumber of a Running H/F variable, as shown below.



In FrameMaker 6/7, you only need to insert the Chapter Number variable (`<$chapnum>`), as shown below.



With the introduction of the Chapter Number variable, you gain back one or more Running H/F variables for more interesting uses.

By inserting a `<$marker1>` or `<$marker2>` building block in a Running H/F, you can extract the content of a Header/Footer $1 or Header/Footer $2 Marker. While you used this trick to insert the hyphen for complex page numbers (see *Using One Master Page for Complex and Simple Page Numbering* on page 158), you can do all sorts of things with this Marker/Building Block combination.

If you had a long section name, longer than would fit in a Running H/F, rather than extract the paratext of that paragraph, you could paraphrase the title in a Header/Footer $1 Marker or Header/Footer $2 Marker, and have the `<$marker1>` or `<$marker2>` building block extract it.

Exercise 142 Using <$marker1> Markers in Running H/Fs

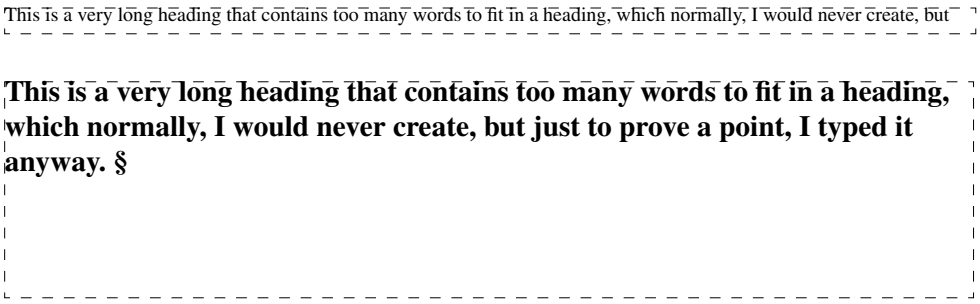
1. Create a blank portrait document. Save it to `../bdme/ch18` as `header-footer-marker.fm`.

2. Go to the master pages. In the Header, add the Running H/F 2 variable.

3. Go back to the body pages and apply the Heading1 paratag to the current paragraph. Type:

   ```
   This is a very long heading that contains too many words to
   fit in a heading, which normally, I would never create, but
   just to prove a point, I typed it anyway.
   ```

4. Refresh the screen (`Control-l` (lowercase L) to see some of the text appear in the header.

   Obviously, the heading is too long to fit in the header.

5. Move your insertion point to the beginning of the paragraph you typed (Windows: `Control-Up Arrow`; UNIX: `Meta-[` [left square bracket])

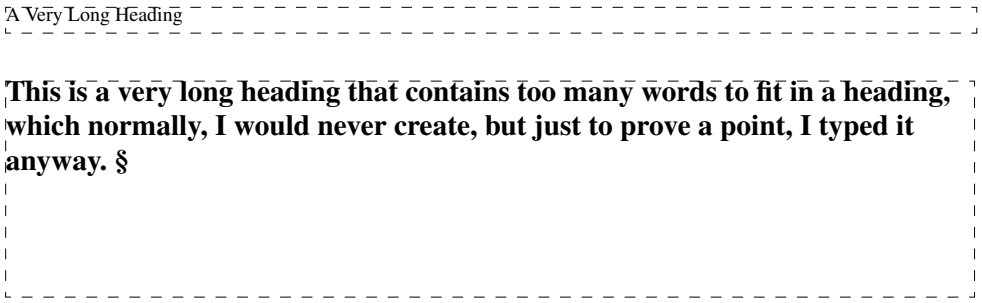
6. Insert a Header/Footer $1 Marker. In the Marker Text box, type: A Very Long Heading

7. Go to the master pages and change the definition for Running H/F 2 to:

   ```
   <$marker1>
   ```

8 Go back to the body pages.

Your header should look like the example shown below:



The only downside to this solution is that unstructured FrameMaker does not have Boolean logic capabilities. That is, you can't have both the <$paratext[Heading1]> and the <$marker1> building blocks in the Running H/F 2 variable because the Running H/F would extract both sets of text. That is, if on a previous page you had a Heading1 paragraph and then inserted a Header/Footer $1 Marker, both the text from the Header/Footer $1 *and* the text from the last previous Heading1 paragraph would appear in the header.

You could create multiple master pages, one with the <$marker1> building block in a Running H/F and another with the <$paratext[Heading1]> building block in the Running H/F; but then you'd have to remember which master page to apply.

The only solution is to split the document up into pieces, so that when you want to have Header/Footer $1 Marker text used in a Running H/F, you would split the document at that point. You'd have no previous Heading1 paratags, and the master pages wouldn't shift because more text was added before the pertinent section.

But all this is probably more work than it is worth. The easiest solution is to make the title shorter.

Landscape Pages

If you need landscape pages in your document, rather than rotating the page, rotate the text frame. Then when you want to globally update the layout for the Left and Right pages, you can do this using **Format > Page Layout > Column Layout**. However, this will give you a rotated text frame, which is very hard to type in. If you must have rotated pages, create them after you have finished modifying any master page layouts.

B Building Blocks in FrameMaker

This is a comprehensive list of all the building blocks that are available in FrameMaker, where they are used, how they are used, what happens when you use them, their power, and their limitations.

System Variables

Chapter Number and Volume Number Variables

New to FrameMaker in v6, the Chapter Number and Volume Number variables can be used anywhere. The lines between the variable itself and its building blocks are blurred, because throughout the rest of this chapter, <$chapnum> and <$volnum> can be used within other variables, and other building blocks can be used within Chapter Number and Volume Number variables.

Table 1: Building Blocks for Chapter Number and Volume Number Variables

Building Block	Definition	Notes
<$chapnum>	The value for the Chapter Number, which is specified in the Numbering Properties window.	One of these building blocks must be present in the Chapter Number or Volume Number variable.
<$volnum>	The value for the Volume Number, which is specified in the Numbering Properties window.	

Table 1: Building Blocks for Chapter Number and Volume Number Variables *(Cont'd)*

Building Block	Definition	Notes
<$curpagenum>	Specifies the current page of the document.	Don't use either of these two building blocks in either the Chapter Number or the Volume Number variable simply because they would limit their use. Presumably, you could add them to create complex page numbers, but that is better handled within the header or footer and chording the Chapter Number or Volume Number variable together with one of the Page Variables (see *Page Variables* on page 645).
<$lastpagenum>	Specifies the last page of the document.	
<$paranum[paratag]>	Extracts the complete autonumber of the preceding paragraph with the paratag indicated in the brackets.	Don't use either of these two building blocks in either the Chapter Number or Volume Number variable simply because the would limit their use.
<$paranumonly [paratag]>	Extracts only the number portion of the autonumber of the preceding paragraph with the paratag indicated in the brackets.	
<*Chartag*>	Format the text following the building block with the character format `chartag`	While you can type in a chartag name before you create it, the chartag must be created in the Character Designer before the characters will display the formatting.

Page Variables

Generally, these variables are used in background, untagged text flows, such as Headers and Footers, but they can also be used on body pages.

Table 2: Current Page# and Page Count Variables

Building Block	How Used
<$curpagenum>	Specifies the current page of the document.
<$lastpagenum>	Specifies the last page of the document. This variable does not count the last page in a book, only the last page in the current document. If you want to count pages in a book, you must create a custom API using FDK or put a marker on the last page of the document and create an x-ref to that marker on the master page. One word of warning: if that marker is moved or text or other files are placed after that marker, it will no longer be accurate. This is a work-around at best. For more information, see *Page Count in Book Files* on page 461.
<$chapnum>	The value for the Chapter Number building block, which is specified in the Numbering Properties window.
<$volnum>	The value for the Volume Number building block, which is specified in the Numbering Properties window. **Note:** Don't use either of these two building blocks in either of the page number variables simply because they would limit their use. Presumably, you could add them to create complex page numbers, but that is handled better within the header or footer and chording the Chapter Number or the Volume Number variable together with one of the page number variables.
<*Chartag*>	Format the text following the building block with the character format chartag. While you can type in a chartag name before you create it, the chartag must be created in the Character Designer before the characters will display the formatting.

How these can be used

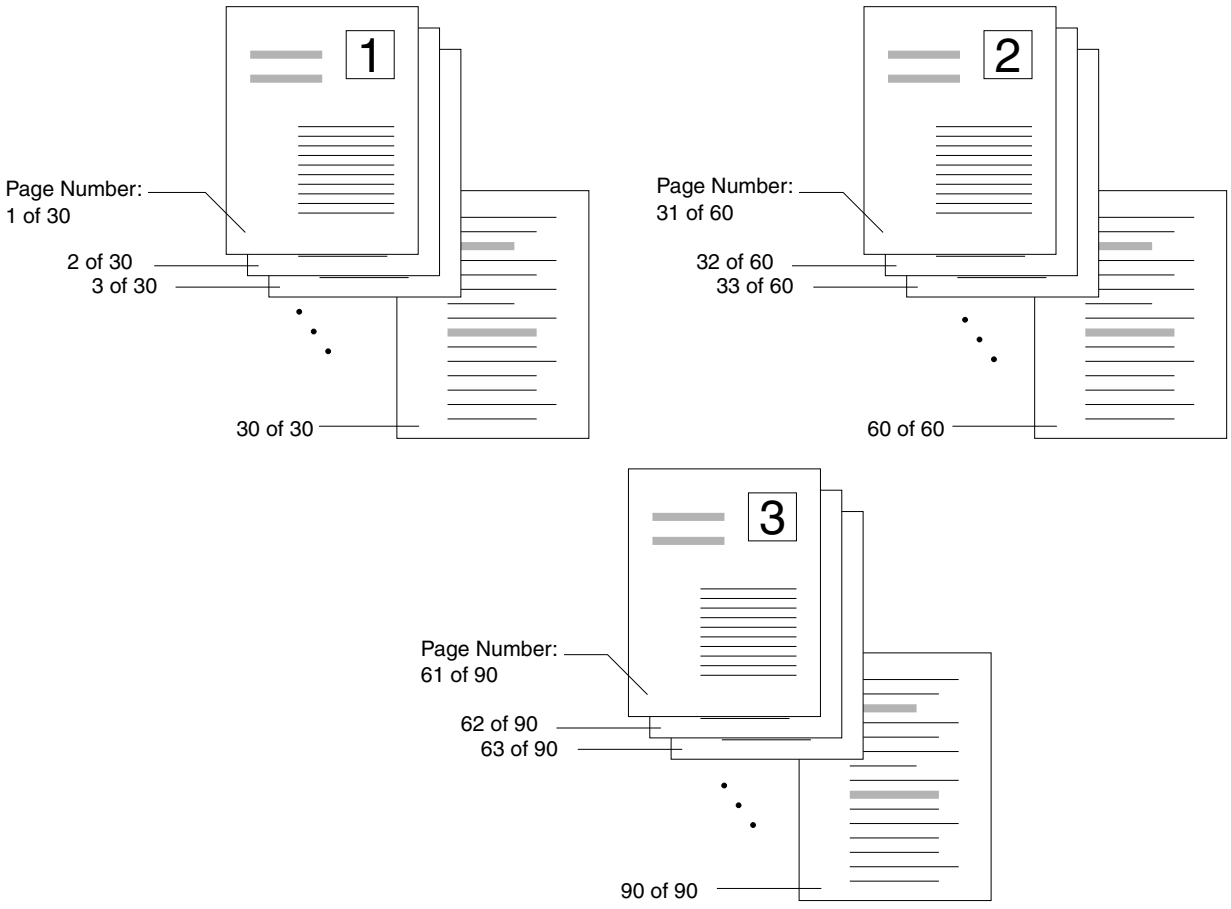
You might prefer to use the default definitions of these two variables. While you could alter the definition of Current Page# to be

```
<$curpagenum> of <$lastpagenum>
```

which would give you the page number and then the last page and would look like this:

```
13 of 16
```

The only problem with doing this is that if you put the files in a book, the <$lastpagenum> building block only counts pages in the current file, not in the entire book. Thus if you had three files in a book, each with 30 pages, the Current Page# output for the files would look like this:



For another solution to displaying a page count in a book, see *Page Count in Book Files* on page 461.

Date Variables

The differences between current, creation, and modification date variables are as follows:

- Current Date shows a dynamic date (based on the current date and time from your OS date and time) according to the building blocks you have configured for that variable
- Creation Date shows a static date and time based on the date the file was created
- Modification Date shows a date based on the last date and time that the file was changed

The spelling of the Date variables is determined by the localization chosen when installing FrameMaker. For example, if English is chosen the Date variable might appear as December 22, 2004, and for German it might appear as Dezember 22, 2004. The spelling shown below is based on an English localization.

Table 3: Current Date, Modification Date, and Creation Date Variables

Building Block	How Used	Example
<$second>	Formats the seconds without leading zero.	7
<$second00>	Formats the seconds with leading zero.	07
<$minute>	Formats the minutes without leading zero	9
<$minute00>	Formats the minutes with leading zero	09
<$hour>	Formats the hours without leading zero	6
<$hour01>	Formats the hours with leading zero	06
<$hour24>	Formats the hours in 24 Hour format	22
<$ampm>	Provides lowercase am or pm	am
<$AMPM>	Provides UPPERCASE AM or PM	PM
<$daynum>	Presents the day of the month as a number, without leading zero.	8
<$daynum01>	As above, but with leading zero.	08
<$dayname>	Presents the day of the week spelled out.	Sunday, Monday, Tuesday, etc.
<$shortdayname>	Presents a three-letter abbreviation of the day of the week.	Sunday = Sun, Monday = Mon
<$monthnum>	Presents the month as a number.	September = 9, December = 12
<$monthnum01>	As above, but with leading zero.	September = 09
<$monthname>	Presents the month spelled out.	March, June, November
<$shortmonthname>	Presents a three-letter abbreviation of the month.	Mar, Jun, Nov
<$year>	Presents the year as four digits.	1776, 1787, 1865, 1969, 2001
<$shortyear>	Presents the year as two digits.	76, 87, 65, 69, 01
<*Chartag*>	Format the text following the building block with the character format chartag. While you can type in a chartag name before you create it, the chartag must be created in the Character Designer before the characters will display the formatting.	

C Shortcuts

Grouped by Menu

Explanation Key

Explanation	Platform and Keystrokes
Indicates a keyboard shortcut can only be used on a Mac	M: Command-w
Indicates a keyboard shortcut can only be used on UNIX	U: Meta-hyphen
Indicates a keyboard shortcut can only be used on Windows	W: Control-f
Indicates a keyboard shortcut can be used on Mac or Windows	M+W: Esc w w w
Indicates a keyboard shortcut can be used on UNIX or Windows	U+W: Control-l (lowercase L)
Indicates a keyboard shortcut can be used on Mac or UNIX	M+U: Control-e
Indicates a keyboard shortcut can be used on all platforms	All: Esc f n
Indicates a key on the numeric keypad	K9
Indicates a function key, not the shifted F key	F10

Some shortcuts use the Esc (Escape) key. To use these shortcuts press and release the Esc key and then press and release each key in succession. For example, the keyboard shortcut to save a file is: Esc f s, which means to press and release the Esc key, then press and release the unshifted f key, and press and release the unshifted s key. Some Esc shortcuts use a shifted letter; for example the shortcut to save all open files Esc f S means to press and release the Esc key, then press and release the unshifted f key, and press and release the *shifted* s key. In the rest of this book, the Esc key is represented by the exclamation point (!).

All shortcuts shown are for FrameMaker v7. Some shortcuts work for earlier versions as well.

File menu

To choose	Shortcut
File > New	All: Esc f n M: Command-n W: Control-n
File > Open	All: Esc f o M: Command-o W: Control-o
File > Save	All: Esc f s M: Command-s W: Control-s
File > Save All Open Files	All: Esc f S
File > Save As	All: Esc f a M: Shift-F7

File menu (Cont'd)

To choose	Shortcut
File > Revert to Saved	All: Esc f r
File > Print	All: Esc f p M: Command-p W: Control-p
File > Print Setup File > Page Setup (Mac)	M: Shift-F8 W: Control-Shift-p
File > Send	W: Esc f m
File > Send All Open Files	W: Esc f M
File > WebWorks Publisher	All: Esc f W
File > Import > File	All: Esc f i f
File > Import > Formats	All: Esc f i o
File > Import > Object	W: Esc f i b
File > Utilities > Compare Documents	All: Esc f t c
File > Utilities > Document Reports	All: Esc f t r
File > Utilities > HTML Setup	All: Esc f t h
File > Utilities > Create and Apply Formats	All: Esc f t f
File > Utilities > Capture	U: Esc f t p
File > Utilities > Keyboard Macros	U: Esc f t k
File > Preferences	All: Esc f P
File > Adobe Online	M+W: Esc w w w
File > Close	All: Esc f c, Esc f q M: Command-w W: Control-w
File > Close All Open Files	All: Esc f C, Esc f Q
File > Quit (Mac) File > Exit (Win)	M: Command-q W: Alt-F4

Edit menu

To choose	Shortcut
Edit > Undo/Redo	All: Esc e u U: Meta-Backspace M: Command-z W: Control-z
Edit > Cut	All: Esc e x M: Command-x W: Control-x
Edit > Copy	All: Esc e c M: Command-c W: Control-c

Shortcuts Sorted Alphabetically by Task

Explanation Key

Explanation	Platform and Keystrokes
Indicates a keyboard shortcut can only be used on a Mac	M: Command-w
Indicates a keyboard shortcut can only be used on UNIX	U: Meta-hyphen
Indicates a keyboard shortcut can only be used on Windows	W: Control-f
Indicates a keyboard shortcut can be used on Mac or Windows	M+W: Esc w w w
Indicates a keyboard shortcut can be used on UNIX or Windows	U+W: Control-l (lowercase L)
Indicates a keyboard shortcut can be used on Mac or UNIX	M+U: Control-e
Indicates a keyboard shortcut can be used on all platforms	All: Esc f n
Indicates a key on the numeric keypad	K9
Indicates a function key, not the shifted F key	F10
Some shortcuts use the Esc (Escape) key. To use these shortcuts press and release the Esc key and then press and release each key in succession. For example, the keyboard shortcut to save a file is: Esc f s, which means to press and release the Esc key, then press and release the unshifted f key, and press and release the unshifted s key. Some Esc shortcuts use a shifted letter; for example the shortcut to save all open files Esc f S means to press and release the Esc key, then press and release the unshifted f key, and press and release the shifted s key. In the rest of this book, the Esc key is represented by the exclamation point (!).	
All shortcuts shown are for FrameMaker v7. Some shortcuts work for earlier versions as well.	

Task	Shortcut
Activate a hypertext command without locking a document	M: Control-Option-click an active area U: Control-right-click an active area W: Alt-right-click an active area
Add (book) > Add Files	All: Esc f f
Add (book) > Index	All: Esc i x
Add (book) > Index of > Authors	All: Esc i o a
Add (book) > Index of > Markers	All: Esc i o m
Add (book) > Index of > References	All: Esc i o r
Add (book) > Index of > Subjects	All: Esc i o s
Add (book) > List of > Figures	All: Esc l (lowercase L) o f
Add (book) > List of > Markers	All: Esc l (lowercase L) o m
Add (book) > List of > Markers (Alphabetical)	All: Esc l (lowercase L) o M
Add (book) > List of > Paragraphs	All: Esc l (lowercase L) o p
Add (book) > List of > Paragraphs (Alphabetical)	All: Esc l (lowercase L) o P
Add (book) > List of > References	All: Esc l (lowercase L) o r
Add (book) > List of > Tables	All: Esc l (lowercase L) o t
Add (book) > Table of Contents	All: Esc t o c
Add a reshape handle and control points	M: Command-Option-click a line, a polyline, polygon, or freehand curve with reshape handles and control points currently displayed U: Middle-click a line, a polyline, polygon, or freehand curve with reshape handles and control points currently displayed W: Control-click a line, a polyline, polygon, or freehand curve with reshape handles and control points currently displayed
Add a word to automatic corrections	All: Esc l (lowercase L) a c
Add a word to the document dictionary	All: Esc l (lowercase L) a d
Add a word to your personal dictionary (Learn)	All: Esc l (lowercase L) a p
Add columns to left of leftmost selected column	All: Esc t c l (lowercase L)
Add columns to right of rightmost selected column	All: Esc t c r
Add rows above top selected row	All: Esc t R a
Add rows below bottom selected row	All: Control-Return W: Control-j
Align objects along Bottoms	All: Esc j b M: Command-Option-Down Arrow W: Control-F3
Align objects along Left sides	All: Esc j l (lowercase L) M: Command-Option-Left Arrow
Align objects along Left/right centers	All: Esc j c M: Command-Shift-c
Align objects along Right sides	All: Esc j r M: Command-Shift-r
Align objects along Top/bottom centers	All: Esc j m M: Command-Option-K7 W: Control-F2
Align objects along Tops	All: Esc j t M: Command-Option-Up Arrow W: Control-F1
Apply a character format by typing the first few unique characters of its tag (until FrameMaker recognizes it) and then pressing Return	All: Control-8 U+W: Esc q c, F8

Shortcuts Alphabetically Sorted

Explanation Key

Explanation	Platform and Keystrokes
Indicates a keyboard shortcut can only be used on a Mac	M: Command-w
Indicates a keyboard shortcut can only be used on UNIX	U: Meta-hyphen
Indicates a keyboard shortcut can only be used on Windows	W: Control-f
Indicates a keyboard shortcut can be used on Mac or Windows	M+W: Esc w w w
Indicates a keyboard shortcut can be used on UNIX or Windows	U+W: Control-l (lowercase L)
Indicates a keyboard shortcut can be used on Mac or UNIX	M+U: Control-e
Indicates a keyboard shortcut can be used on all platforms	All: Esc f n
Indicates a key on the numeric keypad	K9
Indicates a function key, not the shifted F key	F10

Some shortcuts use the Esc (Escape) key. To use these shortcuts press and release the Esc key and then press and release each key in succession. For example, the keyboard shortcut to save a file is: Esc f s, which means to press and release the Esc key, then press and release the unshifted f key, and press and release the unshifted s key. Some Esc shortcuts use a shifted letter; for example the shortcut to save all open files Esc f S means to press and release the Esc key, then press and release the unshifted f key, and press and release the shifted s key. In the rest of this book, the Esc key is represented by the exclamation point (!).

All shortcuts shown are for FrameMaker v7. Some shortcuts work for earlier versions as well.

Shortcut	Task
All: 0 (zero)	Turn off a checkbox
All: 1 (one)	Turn on a checkbox
All: Backspace	Delete previous character
All: Click the frame name in the status bar or open Object Properties (Esc g o) to rename it	Rename a selected reference frame
All: Control-0 (zero)	Insert a variable by typing the first few unique characters of its name (until FrameMaker recognizes it) and then pressing Return
All: Control-4	Apply a condition tag to selected text by typing the first few unique characters of the tag (until FrameMaker recognizes it) and then pressing Return
All: Control-5	Remove a condition tag from selected text by typing the first few unique characters of the tag (until FrameMaker recognizes it) and then pressing Return
All: Control-6	Make the selected text unconditional
All: Control-8	Apply a character format by typing the first few unique characters of its tag (until FrameMaker recognizes it) and then pressing Return
All: Control-9	Apply a paragraph format by typing the first few unique characters of its tag (until FrameMaker recognizes it) and then pressing Return
All: Control-c	Cancel a dialog box
All: Control-d	Delete next character
All: Control-g	Display the Go To Page dialog box
All: Control-h	Delete previous character
All: Control-k	Delete forward to the end of a line
All: Control-l (lowercase L)	Refresh display
All: Control-Return	Add rows below bottom selected row
All: Double click the Word and then Shift-click at the end of the range of words	Select a word, then next words
All: Double-click	Put your insertion point in a text frame you just drew
All: Double-click in the text frame or text line	Deselect a text frame or text line and put the insertion point inside it
All: Double-click the item	Move an item in a scroll list to the opposite scroll list
All: Double-click the tag	Move a condition tag between the In and Not In scroll lists
All: Double-click the tag in the As Is scroll list	Move a condition tag from the As Is to the In scroll list
All: Double-click the word	Select a word
All: Double-click the word in the Correction scroll list in the Spelling Checker window	Replace a questioned word
All: Down Arrow	Move to next line
All: Esc 0 (zero) d	change to first dash pattern
All: Esc 0 (zero) f	Change pattern to first fill pattern (black)
All: Esc 0 (zero) p	Change pattern to first pen pattern (black)