FrameMaker Template Design and Enforcement

Author: Dan Emory

Copyright Notice

Synopsis

This paper describes methods I have developed for designing templates for unstructured documents, and assuring that those documents conform to their templates.

1 Overview

Formatting and tagging consistency is the cardinal prerequisite for preserving in perpetuity the capabilities to globally update document formats/definitions, to accomplish document conversions, to support collaborative authoring, and to reuse or repurpose information.

A template is the most effective tool for achieving this consistency, but the act of creating a template and tossing it over the transom to a group of writers won't do much good unless it is followed up by enforcement.

1.1 Why Do Authors Cheat?

Unfortunately, FrameMaker encourages authors of unstructured documents to use ad-hoc format overrides. In particular, the Format > Font, Format > Size, and Format > Style options, together with the Quick Access Bar, the Context (shortcut) menu, and the Paragraph and Character Designer palettes, constitute an open invitation for authors to override the template.

Here are some of the reasons why authors succumb to FrameMaker's invitation to cheat:

1. **Bad template design**

   If the template itself is inadequate, authors are often forced to cheat in order to properly present the information or to work efficiently. The most common causes of bad template design are:

   1. Failure to fully involve authors and quality assurance people in the template development process.
   2. Failure to adopt template enforcement rules in advance of its design.
   3. Failure to perform a comprehensive Document Analysis
   4. Failure to perform Content Modeling.
   5. Failure to optimize the template for efficiency. Authors confronted with tight deadlines must focus on efficiency. If the template forces them to work in inefficient ways, they'll cheat.

2. **Inadequate training in, and documentation of, proper template usage**

A well-designed template for creating complex information almost always requires formal training of authors in how to use the template efficiently and properly, combined with a hypertexted on-line help document for the template that guides authors in its proper usage. For obvious reasons, the on-line help information should be provided in a hypertexted FrameMaker view-only document that uses the template being documented.

3. **Ineffective Management**

Management is responsible for communicating to authors the need for template adherence, and the consequences of nonadherence. The most effective deterrent to cheating is a quality assurance function that nails authors to the wall by detecting all *ad-hoc* overrides, and confronting them with their transgressions. Another positive result of such quality assurance activities is the identification of template design deficiencies that ought to be corrected.

4. **Author Behavior**

Lazy, incompetent, or defiant authors need to be "reeducated" or terminated. A rigorous quality assurance function (described above) is the most effective method for modifying behavior.

1.2 The Ultimate Solution: FrameMaker+SGML (FM+SGML)

FM+SGML enforces the format and structure rules in structured documents

Although structured document authoring has many advantages, the **ultimate advantage** is that formatting and structure rules are automatically enforced by FM+SGML, and cannot be successfully overridden. By importing the document's element definitions back into itself with "Remove Other Format Overrides" turned on, the EDD's format rules are reapplied, and all overrides, including *ad-hoc* formatting of character strings within paragraphs, are removed. And, because authors are almost completely freed from formatting concerns, they can concentrate solely on structure and content, using only the element catalog and the structure view for guided editing. Inserting an element from the Element Catalog automatically inserts the correct document object, and, if the object contains text, FM+SGML formats the text according to the EDD format rule that is applicable to its structural context.

Improved Productivity

If the Element Definition Document (EDD) is well-designed, the elimination of formatting concerns makes it possible to achieve a quantum leap in authoring productivity, and a substantial reduction in the time that must be devoted to quality assurance.

The same analysis and modeling techniques used to design structured documents are applicable to unstructured ones as well

This paper was created as a structured FM+SGML document, using an EDD and templates of my own design. The Document Analysis and Content Modeling methods used in developing that EDD are adaptable to the design of templates for unstructured documents.

The main purpose of this paper is to show that it is possible (although more difficult and time-consuming) for unstructured documents to at least approach the level of template compliance that is so easily attained with structured FM+SGML documents.

2 Template Enforcement Rules

The rules you intend to enforce need to be drawn up prior to designing the template, because those rules will define the scope of the template. That is, if authors will be prohibited from modifying or adding anything in a particular catalog, then the scope of the template for that catalog must be all-encompassing. Most of the rules enumerated below are stated in their most restrictive form, which requires that the template be all-encompassing with respect to most of the itemized aspects of template design.

Page Layouts

Authors shall not modify, rename, delete, or add to the master page set defined in the template, nor shall they make any modifications to those layouts in the body pages that use them.

Reference Pages

Authors shall not modify, rename, or delete any of the reference pages defined in the template. They may, however, add new reference pages containing document objects (e.g., tables) for the purpose of pasting those objects into body pages.

Color Definitions and Color Views

Authors shall not modify, rename, or delete the color definitions and views defined in the template.

Conditional Text Tags

Authors shall not modify, rename, delete, or add to the conditional text tag definitions defined in the template, unless directed to do so, in which case they shall be changed in all document files in a book file.

Paragraph and Character Catalogs

Authors shall not modify, rename, delete, or add to the tags defined in the template's paragraph and character catalogs. The restriction against modification also means that authors shall not modify the name or formatting of any instance of such paragraph or character tags, with the following exceptions:

1. Character format tags may be applied to text strings or individual characters within paragraph.
2. Column and page break overrides are allowed.

Table Catalog

Authors shall not modify, rename, or delete the table format tags defined in the template's table catalog, or in the table ruling styles contained in the Custom Ruling and Shading dialog. Authors may, however:

1. Add new table formats to the table format catalog.
2. Add new ruling styles to the Custom Ruling and Shading dialog.
3. Apply custom ruling and shading from the Custom Ruling and Shading dialog.
4. Use the Row Format dialog to modify min/max height or start position of a table row.
5. Modify any instance of a table format in the Table Designer, so long as those modifications are only applied to the individual table instance, and not to all instances of that format.

Cross-Reference Catalog

Authors shall not modify, rename, delete, or add to the cross-reference formats defined in the template.

Variable Definitions

Authors shall not rename or delete any user-defined variable definitions in the template, nor shall they add new user-defined variable definitions, nor shall they modify the definition of any user-defined variable that has a fixed value (e.g., a variable definition that produces the company name or a special character). System variables that have a fixed format (e.g., dates,

filenames) in the template shall not be modified.

3 Document Analysis

This process is performed before templates are designed in order to determine the required scope of each template. It is followed by the Content Modeling phase, where the results of this analysis are utilized. Document Analysis is often an iterative process involving the steps described below.

Define the Environment

This step includes:

1. Defining how the documents to be produced from the templates will be created, updated, and distributed. Are they:
   - Retrieved from a database?
   - Printed to hardcopy?
   - Distributed on a CD-ROM?
   - Distributed for on-line viewing?
   - Distributed for internal use, external use, or both?
   - Created in a collaborative authoring environment?
   - Periodically updated, and if so, how often?
   - Likely to reuse some material from existing legacy documents?
   - Likely to be repurposed during their lifetime?
   - Likely to have portions of them reused in future documents?
2. Identifying what standards and policies must be followed, including:
   - Format Standards and Style Guides
   - Structure Standards
   - Content Standards
   - Information Security Policies
   - Review and Editing Policies
   - Revision and Updating Policies
   - Storage and Distribution Policies
3. Identifying the types of authors and others who will use FrameMaker, including any third-party tools/API's that will be needed.
4. Identifying the document users (both internal and external), their information needs, and the tools (e.g., viewing software, printers, CD-ROM drives) that will be required to access the documents.
5. Identifying the fonts that are installed on all authoring platforms. The fonts used in the template must be selected from those available on all platforms.
6. Defining the page layouts, including page size, margins, number of columns, and background objects (e.g., running header/footers and graphics) for each document delivery method.
7. Selecting the font family and font sizes for various paragraph types.
8. Classifying and naming the document types.
9. Predicting document evolution.

Define the Generalized Structure of Each Document Type

In this step you must consider both hierarchy and sequence. Structure diagrams can be used to better visualize hierarchy and sequence, and to assign relationships. It is also important at this stage to identify structural components that are common to many different document types.

Identify and Describe the Required Information Types

This step adds more granularity to the generalized structure of each document type. Focus on information types which have hierarchical or multi-object structure, as well as those which require formatting that is different from ordinary body text. For example, this paragraph, and its title in the sidehead are members of the information type named "ChunkLevel1". Up to four levels of titled chunking can be nested within this information type.

4 Content Modeling

If you were designing an EDD/DTD for structured documents, this is the phase where you would define the elements and their structure rules. For unstructured documents, you do more or less the same thing, but you do it for format tags (e.g., paragraph tags, character tags, cross-reference formats) that will be created in the template's catalogs.

Establish Template Naming Conventions

The naming of things (e.g., paragraph, character, cross-reference, and table tags, master pages, reference pages) in the template's catalogs can have a crucially important effect on authoring efficiency and proper template usage. In complex documents, the required number of paragraph tags can easily run well over 100. Catalogs list their tags alphabetically. Grouping related paragraphs together alphabetically, and using descriptive names, can greatly assist authors in finding and selecting the appropriate paragraph tag. For example, all paragraph tags used for producing Lists might have the prefix "List" in each name. Here is an example of how such paragraph tags might be named so that they all appear together in the paragraph catalog:

- ListBulletFirst
- ListBulletFirstIndent
- ListBulletLast
- ListBulletLastIndent
- ListBulletMiddle
- ListBulletMiddleIndent
- ListNumberFirst
- ListNumberFirstIndent
- ListNumberLast
- ListNumberLastIndent
- ListNumberMiddle
- ListNumberMiddleIndent

Where:

First indicates the first item in a list (it may have more space above, and, in the case of a numbered list, it restarts the autonumbering at 1 or a).

Middle indicates that the item is between the first and last items.

Last indicates the last item in the list (it may have more space below)

Indent indicates that the item is indented so it can be nested under an unindented item in a numbered or bulleted list.

It is also important to establish a naming convention for tags that should not be used by authors (e.g., character tags used exclusively in autonumbering specifications, variable definitions, or cross-reference formats). Such tagnames could, for instance, be prefixed with a # symbol.

Content Modeling Example and Syntax

In many cases, a sequence of paragraphs (and/or other object types) may be required to create a particular information type. For example, consider the Alert information type shown below:

Warning: This is an example of a warning, consisting of an empty paragraph named AlertIconWarning that has a reference frame below containing the icon in the sidehead column, followed by an empty spacer paragraph (named AlertSpacer) in the normal text column (to align the first text paragraph of the warning with the top of the icon), followed by the warning text. The first text paragraph (named AlertTextWarning) has the word Warning in red specified as a prefix in the Autonumbering properties.

If second and succeeding text paragraphs (like this one) are needed within the warning, they require a paragraph tag named AlertText without the autonumber prefix. Following the last text paragraph, paragraph tag AlertEndLine is inserted that has a reference frame above containing a line to separate the alert text from the text that follows it.

For the Alert information type (i.e., Notes, Cautions, and Warnings), the sequence of paragraph tags described above is stated in a *content model*, as follows:

```
((AlertWarningIcon | AlertCautionIcon |AlertNoteIcon),
AlertSpacer, (AlertTextWarning | AlertTextCaution |
AlertTextNote), AlertText*, AlertEndLine)
```

Where:

Open and closing parentheses delimit the beginning and end of connected groups that use a single connector type.

The **vertical bar connector** (|) indicates that any one of the paragraph tags in the connected group can be inserted.

The **comma connector** (,) indicates that the paragraph tags in a connected group must occur in the order given.

The **asterisk occurrence** indicator (*) indicates that the paragraph tag or connected group to the left of it is optional, and can occur more than once.

Note that all nine of the paragraph tags in the content model above have names that begin with “Alert”, so that they will all be grouped together alphabetically in the paragraph catalog.

Other connector and occurrence indicators (not used in the above example) include:

The **ampersand connector** (&) indicates that the paragraph tags in the connected group can occur in any order.

The **plus sign occurrence indicator** (+) indicates that the paragraph tag or connected group to the left of it is required, and can

occur more than once.

The **question mark occurrence** indicator (?) indicates that the paragraph tag or connected group to the left of it is optional, and can occur only once.

The syntax described above is the same as that used to specify the content models/structure rules for elements in DTDs and EDDs.

Content models expressed in the form described above also help to define the formatting properties of paragraph tags. For example, the content model clearly shows which paragraph tags should be assigned Keep With Next or Previous properties in the Paragraph Designer's Pagination Properties.

Content Modeling Guidelines

Use DTD/EDD syntax for content models	The syntax is described below the example on the preceding page.
Assign a unique "container" name to each defined content model	This name would correspond to a container element in SGML that contains no document objects or text, only other elements that contain such objects. In unstructured documents, however, the container does not actually exist as a real object.
Tagnames should identify the type of tag	The content models will be confusing if there is no way to identify the tagname types. One way is to include a prefix to the tagname that describes its type (e.g. P- for paragraph tags, C- for character tags, T- for table tags, V- for variable names, X- for cross-references, CT- for conditional text tags). These prefixes should not, however, be included in the actual template tagnames.
Any content model can contain intermixed tagnames and content model container names	When a container name is included in a content model, it represents the entire content model of that container. At the highest levels of structure, the content models are likely to be made up entirely of container names.
A content model can specify exclusions of certain tagnames in containers	Suppose, for example, that a content model for a particular information type includes the Alert container whose content model is described in the example on the preceding page. Suppose further that warnings are not allowed in that information type. Consequently, the exclusions would specify the two tagnames (AlertIconWarning and AlertTextWarning) that are unique to warnings.
Any document object type can be included in content models	What this means is that content models should not be limited to just body text paragraph tags. They might include anchored frames, marker types, conditional text tags, character tags, footnotes and variable names. Also, content models should be developed to specify the default paragraph tags used in heading, body, and tooting rows of each table format.
Each tagname defined in a content model represents a single immutable format.	The same tagname may appear in more than one content model, provided its format is identical in all content models where it appears.
The content modeling documentation should include descriptions of each container and tagname.	These descriptions should define the purpose of each container and tagname. In the case of tagnames, describe the intended use of the tag, and a description of its formatting requirements.

5 Template Design

Good template design is an art form that requires the knowledge of experienced writers, power users of FrameMaker with a strong understanding of typography and page layout, and inventive, analytical minds. Complex templates designed without the participation of writers are foredoomed.

The description provided here is intended to underscore the complexities of template design

This paper was created from an EDD/template (the EDD itself has nearly 200 pages). The EDD includes 155 element definitions for creating different information types and document objects. Those elements contain over 1000 format rules driven by attribute values and/or element structural context. Over 200 multi-use format change lists in the EDD are put together in numerous combinations to implement the formatting specified in those format rules.

The template's paragraph catalog contains over 100 format-rule-specified paragraph tags, and many of those tags are deployed for multiple uses and contexts that modify their formatting, as specified by format-rule-selected format change lists.

An extreme case is the paragraph tag used in this paragraph. It is specified in a number of different elements, and format rules in those elements can produce well over 100 different formatting variations of it.

If the same template had been implemented for unstructured documents, its paragraph catalog would have to contain well over 500 paragraph tags to duplicate the EDD/template's formatting capabilities.

Unlike an unstructured template, the EDD/template can accommodate a number of different document types, including memos, glossaries, appendixes, text insets, point page revision packages, standalone papers like this one, and multi-file books made up of chapters, in which some of the chapters span two or more files. Additionally, the EDD provides (in the form of attributes) many user-specified formatting options, some of which can significantly alter the appearance of an entire document.

It would require many different unstructured templates to replicate the full range of the EDD's formatting capabilities.

The Rulemaking, Document Analysis, and Content Modeling activities provide a blueprint for designing the template

The adopted template enforcement rules, together with the results of the Document Analysis phase, define the scope and purpose of the template. The results of the Content Modeling phase establish tag naming conventions, and identify the structures and tagnames that must be included in the template's catalogs.

Template design has many masters

The rulemaking, document analysis, and content modeling phases impose a panoply of often conflicting requirements and constraints on template design. But the needs of authors must have equal importance. You must prioritize these conflicting requirements before you can resolve the conflicts.

Design intelligently, and don't cut corners

All of the following lapses will come back to bite you:

- Things that might be needed which you decide to leave out.
- Kludgy, overcomplicated, or inefficient ways of doing things which, on more careful analysis, could be made simpler and/or more efficient.
- Implementing multi-column arrays with tabbed paragraphs rather than tables.

- Using non-postscript fonts, or fonts that are unavailable on some authoring platforms.
- Failing to take into account all of the document delivery modes, and the effects of those different modes on formatting and layout.
- Failing to take into account document evolution, particularly the future likelihood of conversions (e.g., to SGML or XML, or to some new DTP that is superior to FrameMaker).

5.1 The First Things to Do

Step 1. Normally, you should begin by opening a new empty document of the correct page size and page layout.

Step 2. Remove from the new document's catalogs all deletable formats/definitions that are inapplicable.

Step 3. If the new template can utilize some formats/definitions from existing template(s), selectively import those formats into the template.

Step 4. Delete from the new template those formats/definitions/layouts/reference pages/reference frames imported in Step 3 from existing template(s) which are inapplicable.

5.2 Page Layouts

If at all possible, limit the master pages to the default Left/Right pair plus a First master page for the title page. This avoids the inevitable struggle with non-default left/right master page pairs, which don't get auto-updated by FrameMaker.

How many columns?

I firmly believe the best layout for most documents consists of a left-aligned sidehead column plus a single normal text column, as is used in this paper. The sidehead serves as the scanning column for section heads and chunk titles to assist readers in rapidly finding the information they're looking for. The normal text column is narrow enough to make it speed-readable, and wide enough to accommodate most graphics and tables. Your paragraph catalog can include two versions of the empty anchor paragraph(s) used to anchor tables and anchored frames—one that is in the sidehead column and straddles the normal text column, and the other that is in the normal text column. The former anchor paragraph(s) accommodate graphics and tables that are too wide to fit in the normal text column. If landscaped tables or graphics are required, single-column landscaped master pages with no sideheads are required.

Margins waste valuable screen real estate

Margins for hardcopy printing waste valuable screen real estate when documents are viewed on-line. If a printed $8^1/_2$ x 11 document has 1-inch margins all the way around, 37% of the document window is occupied by white-space margins.

The template for this paper specifies left/right margins of 0.25", and top/bottom margins of 0.131" (i.e., from the top page edge to the top of the running header text frame, and from the bottom of the running footer text frame to the bottom page edge). Consequently, the portion of the document window containing white-space margins is reduced from 37% to 8%, permitting a higher zoom setting in the document window while still being able to view the entire width of the document.

Also, the font size for ordinary body text is set to 12 points rather than the normal size of 10 points. The larger font size and a higher zoom setting makes it much easier for authors and editors to read, edit, and manipulate text and document objects without having to zoom in to a setting that prevents the entire width of the page from being displayed. This also makes more screen real estate available outside the document window for dialog boxes and palettes.

In effect, this approach creates what was called "oversized repro copy" in that ancient era before WYSIWYG DTPs, scalable fonts, and high-resolution laser printers. To produce the printed documents, the oversized repro was shot at a reduction to produce final-sized negatives for offset printing. You get the same effect by printing this paper at 83% of full size, which reduces the 12-point body text to 10 points, increases the left/right margins to 0.93", and increases the top/bottom margins to 1.04".

This permits two versions of a document to coexist simultaneously in the same file without the necessity of changing page layouts, font sizes, or graphic and table sizes. In the on-line version (without the superfluous margins) all text, graphics, and tables are 20% larger at a 100% zoom setting than they would be if everything appeared in its final size.

5.3 Variables

Variables are under-used in most templates. Here are some of the ways they should be used:

Fractions

Commonly used fractions (e.g., $^1/_2$ $^1/_3$ $^1/_4$ $^1/_5$ $^1/_8$ $^2/_3$ $^2/_5$ $^3/_4$) can be created by user-defined variables. For example the variable for the fraction $^1/_2$ has the following fixed definition:

```
<FmNumerator>1<Default ¶ Font>/<FmDenominator>2
```

Where FmNumerator and FmDenominator are character formats that specify superscript and subscript respectively, using a font size that is 2 points less than the size of ordinary body text.

Special characters and multi-word phrases

Commonly used special characters (e.g., ©, ™, ®, °, ‡, ±, ≤, ≥, ≠, ≡, Ω) and multi-word phrases can easily be defined by user-defined variables with fixed definitions (the definition specifies the FmSymbol character format for characters requiring the Symbol font, or the FmDingbat character format for characters using the Zapf Dingbat font).

Language Translations

Words and phrases which are problematic for translation can be defined in variables. Then, they only need to be translated once.

Enterprise-Specific Variables

Enterprise-specific information (e.g., enterprise name, mailing address, email address, web site URL) are created as user-defined variables.

Document-Set-Specific Variables

Document-set-specific variables (e.g. document title, copyright date, model number, product name, product nickname, product part number, release date, revision number) should be created as user-defined variables. Note that some of this information often appears in running header/footers, allowing the header/footer information to be globally updated.

Also, since all of the information in this category usually appears in the file containing the book title page, the variables can be assigned their book-wide values in that file, which is then used to update those same variables in all the other files of the book. The update action is accomplished from the book file by choosing **File > Import > Formats**, turning everything off

in the Import Formats dialog except Variable Definitions, and importing the definitions into all files of the book.

5.4 Reference Pages

Reference pages should contain much more than just graphic lines used in the Frame Above/Below properties of paragraphs. The reference pages should also serve as a timesaver toolkit for authors and editors. Here are some examples:

Making it easy for authors to view, use, and update variables

I create a reference page named "Variables", which contains a table of the following form:

Table 1. Variable Definitions

Variable Name	Current Value	Required?	Modifiable?
The variable name	The current value of the variable	Yes or No (indicates whether all instances must use the variable rather than ordinary text)	Yes or No (indicates whether the author is allowed to modify the variable definition)

If the author wants to modify the current value of a variable (and is authorized to do so), (s)he simply double-clicks on the current value in the table, which opens the Variable dialog with the variable already selected. Authors can also copy the variable from the Current Value column to the clipboard, and then paste it into text.

Document date/time stamps

It's often important to know when a document was initially created, who created it, when it was last modified, and who modified it. I create a reference page named "Date/Time Stamp" for this purpose, containing the following information that uses the Filename, Creation Date, and Modification Date system variables:

Date/Time Stamps for File: Filename (Long) variable

Created By: creator's name (typed in)

Creation Date/Time: Creation Date (Long) variable

Last Modification Date/Time: Modification Date (Long) variable)

Last Modified By: modifier's name (typed in)

The Modification Date (Long) variable indicates the most recent date/time that the document was saved.

Note: The only problem with putting the date/time stamp information on a reference page is that you can't print it. An alternative method is to create a special master page (e.g., Date/TimeStamp) having only a background text frame containing the date/time stamp information above.

To print the information, first save the document so as to update the modification date. Then, go to the last page of the document, choose **Special > Add Disconnected Pages**, and select the Date/TimeStamp master page. Now, you can print the page. After it's printed, delete the added page. Finally, close the file without saving it.

Pre-formatted tables

If your documents use table types with prescribed column headings/

widths, custom ruling and shading, and paragraph formats, a pre-formatted table for each type can be a real timesaver. Insert the table in a reference page text flow using the applicable table tag and the prescribed number of columns, heading rows, and footing rows. Then, adjust the columns to the prescribed width, and apply any required straddles and custom ruling and shading. Next, fill in the column names in the heading rows using the prescribed paragraph tags, and apply the prescribed body row paragraph tags to the the cells in the body rows. If the table has a title, type in (at least partially) the table title, and insert the Table Continuation and/or Table Sheet variable(s) at the end. If the table has a footing row, apply the prescribed paragraph format to that row.

Authors can then copy any pre-formatted table from the reference pages, paste it into the body text, and fill in the information required in the body and footing rows.

Repetitively used graphics

Includes icons (e.g., note, caution and warning icons), keycap labels, buttons, etc. In a reference page text flow, place each graphic in a sized anchored frame having the prescribed anchoring position and alignment. Authors can copy any anchored frame (with its graphic) from the reference page and paste it into the body text.

Pre-formatted boilerplate text

Includes standardized notes, cautions and warnings, product liability disclaimers, and anything else that you want to standardize. The text can include variables, which are automatically updated to reflect the current values in each document. Authors can copy any of this boilerplate from the reference page and paste it into body text.

Specification flows for generated lists and indexes

These specifications (separate reference page for each flow, with the reference page names and their text flow names conforming to the FrameMaker naming conventions for generated lists and indexes) make it possible to generate properly formatted lists and indexes from any individual document file, allowing authors to check for anomalies, misspellings in index entries, etc. If the document files are later placed in a book file, the generated files in the book will inherit those specifications the first time they're generated from the book file.

By the Way…

When creating reference frames for objects on reference pages, be sure that the Fill is set to None. This will assure that the empty area of the reference frame doesn't conflict with background objects on master pages (e.g., a shaded overlay to the text, or words such as "Draft" or "Final").

5.5 Paragraph, Character, and Cross-Reference Catalogs

The Fundamental Axioms

These axioms have a single purpose:

To assure that any or all paragraphs formats, character formats, and cross-reference formats in any document created from the original template can be changed in the original template (or a surrogate of that template), and that those changes can then be globally propagated successfully into all affected documents.

Axiom 1: All *ad hoc* format overrides (except page and column breaks in paragraphs) are illegal.

This means that the catalogs must have a format for each variation required for the proper presentation of the various information types in any anticipated context, else format overrides must be used.

Axiom 2: Authors cannot change the format of any tag in the paragraph, character, and cross-reference catalogs.

Such changes (which may be significant) will be removed whenever the template is re-imported into the document with Remove Other Format Overrides turned on.

Axiom 3: Authors cannot add new tags/formats to the paragraph, character, and cross-reference catalogs.

Obviously, if such tags/formats were added, they could not be globally updated from the template.

Axiom 4: All format overrides to character strings within paragraphs must be accomplished by the application of the appropriate character tag.

If this axiom is not followed, the resulting overrides cannot be globally changed by updating character formats, nor can they be easily found and removed.

Axiom 5: Character formats must have all properties set to “As Is” except for those properties that are deliberately intended to override the default font properties of **all** paragraphs to which they can be applied.

Although this axiom is self-evident, it is often overlooked. Frequently, for example, the font family or font size is not set to “As Is”, even though there is no intention to change those default properties of the paragraph. Subsequently, if the font family or font size of the paragraph format is globally changed, the formatted string will have the wrong font or font size.

What these axioms mean to the format design process

The template design must make enforcement of the axioms possible in a real-world setting where tight deadlines and budgets, writer turnover, and indifferent attitudes are normally occurring obstacles to enforcement. The format design must forestall both the need and the temptation to violate the axioms.

A poor format design that fails to anticipate all required formatting variations in all possible contexts, fails to satisfy design requirements, or impedes authoring efficiency will assuredly result in rampant violation of those axioms.

Design Pointers

Here is a list of some things to consider.

- **Use descriptive tagnames** (see *Establish Template Naming Conventions in Section 4* for details).
- The value of the **Next ¶ Tag** in the Paragraph Designer Basic Properties panel as a method of improving authoring efficiency is often overlooked. The Content Modeling phase shows instances where one paragraph is always (or usually) followed by a particular paragraph tag that is different. Specifying the next tag as part of such a paragraph format allows authors to just hit the ENTER key to insert the correct (or

most likely) tag for the next paragraph.

- Inserting **repetitively used graphics and/or text** by specifying a Frame Above/Below in the paragraph format. I don't mean just graphic lines. You can also create reference frames on reference pages containing graphics, text, tables, or a mixture of all of them. To insert these objects, you create special paragraph tags for that purpose.

By the Way…

Text insets that are created in individually named text flows of a FrameMaker "fragment" file can serve the same purpose. They can be imported By Reference or By Copy, and options are provided to preserve the original formatting, or to reformat according to the target document's formatting. The advantage of importing text insets by reference is that the documents containing them are automatically updated to reflect the latest version of the source (in the fragment file).

Note also that a fragment file can contain an entire "library" of such text insets, each in a separate, descriptively named text flow.

- Design **paragraph formats** so they'll work in both **body text and table cells**. The Table Cell Properties panel of the Paragraph Designer lets you set up the special properties of paragraphs when they are inserted in table cells, without affecting their properties when they are inserted in body text.
- Decide whether paragraphs should have **fixed or variable line spacing**. If paragraphs can contain footnote numbers, fractions where the numerator and denominator are superscripted, or other instances of subscripts or superscripts, you'll have to decide which line spacing option (Fixed or Variable) is best. Even when you select Fixed line spacing, the amount of spacing is adjusted to reflect any changes in the font size. Remember that the amount of spacing depends not only on the font size, but also on which option (Single, 1.5, or Double) you select in the Line Spacing menu.

By the Way…

You can create two versions of your templates—one for draft and one for final, where the only difference in the two templates is the line spacing setting (e.g., space-and-a-half for draft, single for final). To produce the draft version of a document, simply import the paragraph formats from the draft version of the template, with format overrides turned on. Then, to produce the final version, simply repeat that process, using instead the final version of the template.

- Although there will be exceptions, you should avoid specifying a non-zero value in both **Space Above and Space Below** in the Basic Properties panel of the Paragraph Designer. It is best to specify only a Space Above. Obviously, there will be exceptions to this rule, but they should be minimized so that the spacing will be the same in all contexts. If you do specify a Space Below, it should have a larger value than the Space Above setting of the paragraph that usually follows it, otherwise it will be overridden by the Space Above specified in the

following paragraph. This approach will minimize the number of cases where multiple paragraph tags for the same basic paragraph type must be created simply to accommodate variations in Space Above/ Below values for different contexts.

By the Way…

A special paragraph tag (let's call it PageBreak) can be added to the Paragraph Catalog to produce page or column breaks anywhere in a document by setting the Space Below in that paragraph format to a value large enough to always force the next paragraph to appear at the top of the next column or page.

You simply insert PageBreak (an empty paragraph) as the last paragraph in a page or text column. This eliminates the need to apply Page Break or Column Break overrides to other paragraph formats in the Paragraph catalog.

If you want to move a page or column break to another position, you simply cut and paste the empty PageBreak paragraph.

- Remember that you can apply **character formats in combination** to the same string. If you have a character format named Bold, and another named Italic. you can apply both to the same string, thus there is no need to clutter up the character catalog with combinations such as Bold-Italic, unless that particular combination has a specific purpose.
- **Character formats can be used for things other than changing the format of a string**. That is, you can create character formats with descriptive names that have all properties set to "As Is". The purpose of such tags is to identify something you may want to search for. For example, suppose you create an all-As-Is format named PartNumber to tag all part numbers appearing in the text. Now, you can search on that character tag and find all instances of part numbers in the text. Later, if you decide to make all part numbers stand out from ordinary text, you can globally update the
  PartNumber character format to change one or more of its properties.
- You may also want to create **character formats that are used exclusively in variable definitions and cross-references** to apply distinctive colors to those objects so they will stand out from ordinary text. Before you print to hardcopy, you can globally change these formats back to black, then restore the colors when authoring resumes. Special character formats that are used exclusively in **autonumbering specifications** may also be required.

  The names of such special character tags should include a distinctive prefix (e.g., #) which identifies them as tags which are not to be used by authors to format ordinary text strings within paragraphs. This will assure that you can globally update those formats without affecting anything other than the objects they were intended to format.

5.6 Surrogate Templates

Surrogate templates provide a way to adapt the master template for use with a particular document set. After importing formats from the master template into the document set, the modified formats/definitions in the surrogate template are imported into the same document set, thereby adding to, and/or overriding, the formats in the master template. The result is that the document set's format catalogs and definitions are a composite of the formats/definitions from the master template, and those formats which are subsequently imported from the surrogate template.

In some cases, all of the formats/definitions in a particular catalog are imported from the surrogate. In that event, the particular format/definition category is turned off when importing the master template into a document set, so that the catalog's contents are completely determined by the subsequent importation of the surrogate template.

How are they created?

A surrogate template is created by cloning and renaming the master template. Then, format tags/definitions which are not applicable to the surrogate template's purpose are deleted from the catalogs, leaving only those format tags/definitions which need to be modified for a particular document set. After making the required modifications, it may also be necessary to add new formats/definitions needed for that particular document set.

Typical ways they are utilized

1. Document-set-specific variables (see *Document-Set-Specific Variables in Section 5.3*), as well as the Current Page # system variable, often deviate from the generic values for these variables in the master template. The correct values for these variables can be defined in a surrogate template.
2. New variable definitions applicable only to a particular document set can be defined.
3. New conditional text tags applicable only to a particular document set can be defined.
4. Changing a document's page size, or other page layout parameters.
5. Adding new reference pages not included in the master template's reference page set.
6. Changing the formats for some or all of the tags in the paragraph and character catalogs.
7. Adding paragraph and character formats that are only associated with a content model that is not implemented in the master template.
8. Modifying or adding to the color definitions, cross-reference formats, or table formats.

6 Testing and Updating the Templates

I always create two versions of a template, one (the master template) is empty. The other is a sample document which contains the latest version of the master, and has instantiations of all the content models that were defined during the Content Modeling phase. That means it should contain all document object types, including page layouts, table formats, graphics, variables, and cross-reference formats, as well as paragraph and character formats in all possible context variations. All required or proposed template modifications are first tested in the sample document.

Additions and modifications are first refined in the sample document. Then, the master template is updated by importing formats from the sample document into the master.

New Information Types

Each new information type that is identified should first be defined by a content model. The content model will identify any new formats and object types that must be added, as well as existing ones that already appear in other content models. An instantiation of the the new information type is then created in the sample document. Any required new formats are created and refined, and the impact, if any, of these changes on other content models should be evaluated.

Revised Content Models

Whenever an existing content model is changed, it must be changed in the instantiation(s) of it in the sample document. The change is then analyzed to determine whether any format modifications or additions are required. If modifications to existing formats are made, the impact of these changes on other content models that use the modified format tags must be evaluated to determine whether they are compatible with all content models where the modified format tags are used.

If the content model modification involves the deletion of a particular format tag, and that tag is not used in any other content model, it should be deleted from the catalog in both the sample document and the master template.

Suggested Changes to Existing Formats

First, search the sample document for all instantiations of the format that is to be changed, and determine whether the proposed change is compatible with all instantiations of it. If the suggested change is valid, but it is incompatible with some instantiations, the change may require the addition of new formats.

7 Producing Paragraph Listings

MIFMuncher is a shareware product that can produce the complete details of each paragraph format in the paragraph catalog. First, you save the template in MIF format. Then, run MIFMuncher on the MIF file to produce an ASCII file containing the format details of each paragraph. You can then open the ASCII file in FrameMaker.

You can also use MIFMuncher to produce a simple list of all paragraph tags in the catalog, without their formatting details. By generating such a list for the template, you can compare it to the corresponding list produced for each document created from the template. This allows you to identify illegally added paragraph tags in documents created from the template.

8 Safeguarding Templates

The master copies of templates, as well as their companion sample document version(s) must be protected from font substitution and other random acts of violence committed by users, the software, and the computer system.

Back up your work

Each time you modify a template or its companion sample document, back it up to a Zip disk or similar media where it is protected from hard disk crashes.

Protect the Template from Corruption by the Sample Document

Each time you open a sample document, the first thing you must do is to import into it the formats from its companion master template with Remove Other Format Overrides turned on. This assures that the sample document is initially in sync with the master before any changes are made in the sample document. Each time you make a modification to the sample document, and before you update the master from the sample document, be sure to import the sample document's formats back into itself (i.e., select Current as the document to import the formats from) with Remove Other Format Overrides turned on.

Protect templates against font substitution

Font substitution is the equivalent of the AIDS virus. There are two carriers:

1. **Win 9x and Win NT platforms**: If your template uses Postscript fonts, and a non-postscript printer is selected before or while the master template or the companion sample document is open, abominable TrueType fonts will be substituted.
2. **Any platform that is missing fonts used in the template**: Opening the template or the companion sample document on such a platform will cause font substitution for the missing fonts to occur.

The safest approach is never to allow either of these events to happen. Always check the Console Log immediately after opening a template or a sample document to make sure there were no font substitutions. If font substitutions occurred, immediately close the file without saving it. In FrameMaker version 5.5.x and above, the Ghost fonts feature can restore the original fonts when the cause of the font substitution is eliminated. This, however, requires that you always turn on Remember Missing Font Names in the Preferences dialog before opening the template or its companion sample document. Note, however, that turning on Remember Missing Font Names after opening the document will preserve the substituted font names rather than the original font names.

Prevent write access by unauthorized persons

Templates and their companion sample documents must reside in directories where write access by unauthorized persons is denied.

9 Template Enforcement

The objectives of a template are to achieve consistent tagging and formatting. Template enforcement involves the necessary quality assurance steps that must be performed on each completed document instance to assure that those objectives are achieved.

9.1 Style Guide

Any template worth designing deserves a style guide. The best approach is to utilize the sample document (see Section 6, *Testing and Updating the Templates*) as a starting point. Add explanatory text plus content models for each information type. Finally, add index markers, and generate a hypertexted index, plus a hypertexted Table of Contents. Make the style guide available as a view only on-line document to all authors and quality assurance personnel.

9.2 Tools of the Trade

The following tools are useful for template enforcement:

MIFMuncher

See Section 7, *Producing Paragraph Listings*.

Create and Apply Formats

How is it invoked?

With input focus in the document being analyzed, choose **File > Utilities > Create and Apply Formats**.

What does it do?

It scans a document, and finds all occurrences of:

- **Format overrides to paragraphs**. Each such paragraph is assigned a new tag, which is added to the paragraph catalog. For example, if there are three instances where properties of the Body paragraph format are overridden, three new paragraph tags, Body1, Body2, and Body3 are added to the paragraph catalog
- **Untagged formatted strings within paragraphs**. Each such string is assigned a new tag, which is added to the character catalog. For example, if three strings in different paragraphs have an ad-hoc Bold property that overrides the default non-bold property of the paragraph format, all three strings would be tagged with a new character format named CharFmt. Then, if another such string is found that has an Italic property that overrides the default non-italic property of the paragraph format, it would be tagged with a new character format named CharFmt1. Both of these new tags would be added to the character catalog.

What are its limitations

1. It finds paragraphs where the only override is that Page Break or Column Break has been turned on, and assigns them new paragraph tags. Since page and column breaks are not normally treated as format overrides, these new tags are usually superfluous.
2. It cannot be used on an entire book. Instead, it must be performed on each file in the book.

What is its purpose in template enforcement?

1. It allows the scope of the override problem to be determined.
2. It prevents overrides from being wiped out if the template's paragraph and character formats are subsequently imported into the document with Remove Other Format Overrides turned on.

3. It provides the means (using a hypertexted generated list of paragraphs) to locate each instance of each new tag that was created, so that the format overrides in each instance can be evaluated before determining of how they are to be fixed.

Generate a Hypertexted List of Paragraphs

How is it invoked?

With input focus in the document whose paragraphs are to be listed, Choose **File > Generate/Book**. In the Generate/Book dialog, choose List of Paragraphs from the List menu, and click the Generate button. Then, in the Set Up List of Paragraphs dialog, move the paragraph tags of interest into the Include Paragraphs listbox, turn on Create Hypertext Links, and click the Generate button. The list of paragraphs will then be generated and displayed.

What does it do?

The text of each paragraph whose format tagname matches the tagname of a paragraph listed in the Include Paragraphs listbox is produced. Each paragraph has a format tag named tagnameLOP, where tagname is the name of a tag that was included in the Include Paragraphs listbox. Each paragraph in the generated list has a hypertext marker. After the generated list is made View Only, clicking in a paragraph produces a hypertext jump to the actual paragraph in the document.

What is its purpose in template enforcement?

It provides a way to list, find, evaluate and apply the correct paragraph tag to, all instantiations of paragraphs whose tagnames are not included in the template (i.e., paragraph tags illegally added to the paragraph catalog, plus paragraph tags for format overrides created by executing the Create and Apply Formats action).

Generate a Hypertexted List of Fonts

How is it invoked?

With input focus in the document whose fonts are to be listed, Choose **File > Generate/Book**. In the Generate/Book dialog, choose List of References from the List menu, and click the Generate button. Then, in the Set Up List of References dialog, move Fonts into the Include References listbox, turn on Create Hypertext Links, and click the Generate button. The list of fonts will then be generated and displayed.

What does it do?

Each font occurring in the document is listed, and has a hypertext marker. After the generated list is made View Only, clicking in a font description produces a hypertext jump to the instance of the font in the document.

What is its purpose in template enforcement?

It provides a way to list, find, evaluate and fix all instantiations of fonts that are not included in the template.

Find a Character Format

How is it invoked?

With input focus in the document in which character tags are being examined, choose **Edit > Find/Change**. In the Find/Change dialog, choose Character Format from the Find menu, and type in the name of the character format to be found in the slot to the right of the Find menu.

What does it do?

It finds all instantiations of the specified character format.

What is its purpose in template enforcement?

It provides a way to find, evaluate and fix all instantiations of character formats whose tagnames are not included in the template (i.e., character tags illegally added to the character catalog, plus character tags for format

	overrides created by executing the Create and Apply Formats action).
Find a Cross-Reference Format	
How is it invoked?	With input focus in the document in which cross-reference formats are being examined, choose **Edit > Find/Change**. In the Find/Change dialog, choose Cross-Reference of Format from the Find menu, and type in the name of the cross-reference format to be found in the slot to the right of the Find menu.
What does it do?	It finds all instantiations of the specified cross-reference format.
What is its purpose in template enforcement?	It provides a way to find, evaluate and apply the correct cross-reference format to, all instantiations of cross-reference formats whose tagnames are not included in the template (i.e., cross-reference formats illegally added to the cross-reference catalog).
Find a Variable	
How is it invoked?	With input focus in the document in which definitions are being examined, choose **Edit > Find/Change**. In the Find/Change dialog, choose Variable of Name from the Find menu, and type in the name of the variable to be found in the slot to the right of the Find menu.
What does it do?	It finds all instantiations of the specified variable.
What is its purpose in template enforcement?	It provides a way to find, evaluate and fix all instantiations of variable definitions which are not included in the template (i.e., variable definitions that were illegally added to the document).
Find a Conditional Text Tag	
How is it invoked?	With input focus in the document in which conditional text tags are being examined, choose **Edit > Find/Change**. In the Find/Change dialog, choose Conditional Text… to open the Find Conditional Text dialog. In this dialog, turn on Conditional, move the conditional text tags of interest into the In listbox, and click the SET button.
What does it do?	It finds all instantiations of the specified conditional text tags.
What is its purpose in template enforcement?	It provides a way to find and evaluate all instantiations of conditional text tags which are not included in the template (i.e., conditional text tags that were illegally added to the document).
Find Unresolved Cross-References	
How is it invoked?	With input focus in the document in which unresolved cross-references are being examined, choose **Edit > Find/Change**. In the Find/Change dialog, choose Unresolved Cross Reference from the Find menu.
What is its purpose in template enforcement?	It provides a way to find and fix unresolved cross-references.

Import Formats

How is it invoked?	With input focus in the target document, choose **File > Import Formats**. In the Import Formats dialog, choose the source document you want to import the formats from (it must already be open). Next, turn on the formats you want to import. The format selections include: 1. Paragraph formats 2. Character formats 3. Page Layouts (i.e., master pages) 4. Table Formats 5. Color Definitions 6. Reference Pages 7. Variable Definitions 8. Cross-Reference Formats 9. Conditional Text Settings 10. Math Definitions Finally, select whether you want to remove format overrides.
What does it do?	The selected format types from the source document update the correspondingly named formats in the target document, and any new tags not already present in the target document are added. If Remove Other Format Overrides is turned on and Remove Manual Page Breaks is turned off, format overrides (except page and column breaks) in all instantiations of the selected format types within the target document are removed.
What are its limitations?	If there are formats in the target document whose tagnames do not match any of the tagnames in the source document, those formats and all their instantiations are unaffected. Also, if Table Formats are imported with Remove Other Format Overrides turned on, legitimate overrides in instantiations of tables created from those formats will be removed.
What is its purpose in template enforcement?	To bring the target document's formats into conformance with selected formats from the template.

9.3 Enforcement Procedure

This procedure can be used to bring any document into conformance with the template from which it was created. The methodology assures that all format overrides can be properly evaluated before they are removed.

Note: In steps 1 and 2 below, you can ignore Table Formats if authors are allowed to create new table formats (see *Table Catalog in Section 2*).

Step 1. Make a list of all formats/definitions/page layouts in each of the categories described under *Import Formats in Section 9.2* for the master template, plus any additional ones in applicable surrogate templates. In the case of paragraph formats, you can use MIFMuncher (see Section 7, *Producing Paragraph Listings* to produce the list.

Note: Deletion of tags/definitions from the catalogs in steps 2 and 3 below does not affect any instantiations of those tags/definitions (i.e., they will still retain the tag, even though the tag/definition has been deleted from the catalog).

Step 2. Open the document whose conformance to the template is to be enforced, and make a list of all formats/definitions/page layouts as in step 1, and compare the two lists. Make a list of all tags/definitions/page layouts that have been illegally added to the document, then delete those tags (except for variable definitions) from the document's catalogs/definitions/page layouts.

Step 3. Perform the Create and Apply Formats action (see *Create and Apply Formats in Section 9.2*), and identify and list all new paragraph and character formats that were added by this action. Then, after making the list, delete all of those new paragraph and character tags from the catalogs.

Note: The actions performed in Steps 4 and 5 below will not override paragraph and character formats identified in Steps 2 and 3, because formats for these tags are not included in the template(s).

Step 4. Open the master template, and perform the Import Formats action (see *Import Formats in Section 9.2*) to import all formats (except for Table Formats and catalogs that are completely defined in the surrogate template) from the master template, with Remove Other Format Overrides turned on.

Step 5. If a surrogate template is also being used (see Section 5.6, *Surrogate Templates*), open it and perform the Import Formats action again to import all applicable formats except Table Formats from the surrogate template with Remove Other Format Overrides turned on.

Note: Be sure that Remove Other Format Overrides is turned off in Steps 6 and 7 below, otherwise, customizations of table instances (e.g., ruling and shading, changes in cell margins) will be wiped out.

Note: Do not perform Step 6 below if all table formats are being imported from the surrogate template.

Step 6. Perform the Import Formats action again with only Table Formats selected to import the Table Formats from the master template with Remove Other Format Overrides turned *off*.

Step 7. If a surrogate template is also being used, and it contains Table Formats, perform the Import Formats action again with only Table Formats selected to import the Table Formats from the surrogate template with Remove Other Format Overrides turned *off*. After this step is completed, all templates can be closed.

Note: At the completion of step 7, the document's catalogs will contain only the formats/definitions/page layouts imported from the templates, and all instantiations of those formats will conform to the formatting specified in the templates. However, instantiations of deviant paragraph, character and cross-reference formats, as well as deviant variable definitions and conditional text tags (all of which were identified in steps 2 and 3), are still present, and are still non-conforming.

Step 8. You must now use the lists (produced in Steps 2 and 3 above) of deviant paragraph, character and cross-reference formats, deviant conditional text tags, and deviant variable definitions to find and fix them. The methods described in Section 9.2 that are used to find them include:

1. Generate a Hypertexted List of Paragraphs
2. Find a Character Format
3. Find a Variable
4. Find a Cross-Reference Format
5. Find a Conditional Text Tag

Step 9. Update the cross-references, and fix unresolved ones. Choose **Edit > Update References**, turn on All Cross References in the Update References dialog, and click the Update button. If unresolved cross-references are found use *Find Unresolved Cross-References in Section 9.2* to find and fix them.

Step 10. Check for invalid fonts, using the method described under *Generate a Hypertexted List of Fonts in Section 9.2*, and change any instances of invalid fonts to the correct ones.

Step 11. The document is now fully conformant with the template(s). In the process of doing that, you probably found that some paragraph and character format overrides were the result of applying the wrong tag. Instead of changing the tag to the appropriate one, the author applied format overrides to the paragraph or character format. There may, however, still be cases where the inappropriate tag was applied, without any format overrides being used. You should now scan through the document for improper tag usage, and apply the correct tag to anything that's found.

Dan Emory & Associates

Information Design Specialists

Dan Emory

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com

Dan Emory & Associates

Information Design Specialists

Dan Emory

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com

Dan Emory & Associates

Information Design Specialists

Dan Emory

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com

Dan Emory & Associates

Information Design Specialists

Dan Emory

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com

Dan Emory & Associates

Information Design Specialists

Dan Emory

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com

Dan Emory & Associates

Information Design Specialists

Dan Emory

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com

Dan Emory & Associates

Information Design Specialists

Dan Emory

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com

Dan Emory & Associates

Information Design Specialists

Dan Emory

10044 Adams Ave. #208
Huntington Beach, CA 92646
Voice/Fax: 949-722-8971
Email: danemory@primenet.com