

FrameMaker Generated Files

Adobe's FrameMaker product ("Frame") has powerful facilities to generate and update book components such as Table of Contents, List of Tables, List of Tables, and Indices. However, these facilities are complex. This document provides an outline of the technical aspects of the *Generate* and *Update* operations. I describe Frame version 7.0p577 for Macintosh. The principles are unchanged in more modern versions. I assume that you're quite familiar with using Frame to edit ordinary documents, and I assume loose familiarity with Frame *books*.

Book files

Non-document files, even book files, can be included in a book as a convenience to the writer/editor: Double-clicking such an entry uses the Finder to open the associated file external to Frame.

A *book* in Frame is an ordered collection of Frame document files. A Frame document in a book can be an *ordinary* file, placed in the book using *Add > Files ...*. Alternatively, a book component can be a *generated* file such as a Table of Contents.

When placed in text, cross-references in Frame function atomically: When you click on a cross-reference, the whole cross-reference is selected; it can be Cut, Copied, Pasted, deleted, etc. Also, the content of a cross-reference is subject to dynamic updating (by *Edit > Update References ...*, for example). Content in generated files is not like that: The *Generate* process copies and formats text into the generated file; once placed, such text can be manually edited. Any such edits are liable to be overwritten the next time *Generate* is performed. Such editing can be expedient for last-minute touch-ups, but ordinarily you'll want to edit the *source* of information in the generated text, not its instantiation in the generated file. Ordinarily, you will want to maintain a book in a state such that *Update Book* can be applied at any time without disturbing essential content.

Book from scratch

Creating a book from scratch is simple: Merely choose *File > New Book ...*, then add ordinary files (*Add > Files ...*).

It is feasible to create generated files from scratch using the other entries in the *Add* menu such as *Add .> Table Of Contents ...*, but expect to spend significant time adjusting the formatting of these files.

You can add a generated file (such as a pre-existing TOC file) using *Add > File ...*, but doing so doesn't imbue the file with the any *Generate/Update* properties.

Book from a prototype

When faced with the task of creating new generated files, the easiest approach is to start within existing book and document files that can serve as a prototype.

Frame documents often contain references to other Frame documents in the form of text insets, cross-references, etc. Paths to such files are stored in the document in relative form, not absolute. The easiest way to copy a Frame file keeping the links functional is to open it within Frame and then *Save As ...*

When a book file is frontmost in Frame, you can *Save As ...* to save a copy functionally equivalent to the original book. When you perform this act on a book file named *EXAMPLE.book*, and save as *NEW.book*, generated files in the previous book having names such as *EXAMPLE.TOC.fm*, *EXAMPLE.LOF.fm*, *EXAMPLE.LOT.fm*, and *EXAMPLE.IX.fm* will be renamed *NEW.TOC.fm*, *NEW.LOF.fm*, *NEW.LOT.fm*, and *NEW.IX.fm*. That renaming is done *within the book*, without reference to whether any such files exist on disk. (If the corresponding new file doesn't exist, an attempt to open it will fail.) This mechanism allows an expedient way to copy a previous book's structure to a new one: Use the Finder to copy *EXAMPLE.TOC.fm* to *NEW.TOC.fm*, and similarly for each generated file; then save the previous book file to *NEW.book*. You'll find it most convenient to keep a book file and all of its components in a single directory.

Update Book operation

Update Book ... is used to scan all of the components of a book, extract relevant elements for generated files, format the elements, and sort those elements if necessary. Invoking *Update* potentially performs individual functions or multiple functions. As mentioned previously, you will typically avoid direct editing of generated content in order that *Update Book* can be applied at any time without overwriting any edits that you have performed.

Update Numbering

The *Update Numbering* option of *Update Book* scans all of the component files of a book to renumber pages and to propagate and/or increment volume, chapter, and paragraph numbers.

Each document carries properties that give the page number style (numeric, alphabetic, Roman numeral, etc.) and the starting page number. These properties can be changed in the document itself (*Format > Document > Numbering ...*). However, page numbering properties for each component of a book are carried in the book as well, and problems arise if the properties aren't matched. If numbering properties are to be changed, it is advisable to change them from the book file, not from the document.

Each document carries an invisible property that gives a starting paragraph autonumber (as a list). There is no user interface to this property; it is set by the *Update Numbering* operation.

In versions of Frame prior to version 7, chapter numbering was accomplished by incrementing a component of the paragraph autonumber. The Chapter paragraph would have an autonumber such as `<n+><=0><=0><=0><=0><=0>` which incremented the chapter number component and reset all other autonumber components to zero. Autonumbers were propagated across book components.

Frame 7 introduced volume and chapter number mechanisms that don't rely upon paragraph autonumber formats. It is recommended to use the new mechanism: Your Chapter paragraph will carry an autonumber such as `<$Chapter>`. Rather than including the autonumber building block `<=0>` to reset other counter elements, the *Format > Document > Numbering* properties will be set to *Reset Paragraph Numbering*.

Update All Cross-References

When *Update Book ...* is invoked, it is sensible to update all cross-references; otherwise, if page numbers, paragraph numbers, cross-referenced paragraph text or any other elements of cross-reference sources have changed, those changed will not be reflected in the new book.

Update All Text Insets

When *Update Book ...* is invoked, you may choose to update all text insets. Whether or not you do so is a matter of your own workflow and procedures.

Generate

Generate is the action of extracting, collating, sorting, and formatting certain elements of book components, and inserting the generated content into generated files.

The elements that are scanned for inclusion in generated content intended for inclusion in a publication are paragraphs and markers. For use in document production tasks, Frame provides a mechanism to scan *references* to elements such as fonts, text insets, and imported graphics. Generated files containing these elements

Two kinds of processing are implemented. *Lists* are simply collated lists of elements, placed in the generated file in order of occurrence in the book. *Indices* are sorted lists of elements, placed in the generated file according to the alphabetization of the paragraph text or the marker text elements.

Table 1 summarizes the various possibilities.

<i>Default suffix</i>	<i>Derive type</i>	<i>Name</i>	<i>Source</i>	<i>Default Marker type</i>	<i>Ordering</i>	<i>Building blocks</i>
GENERATED LIST						
TOC	TOC	Table of Contents	Paragraph		Collate	
LOF	LOF	List of Figures	Paragraph		Collate	
LOT	LOT	List of Tables	Paragraph		Collate	
LOP	LOP	List of Paragraphs	Paragraph		Collate	
APL	APL	Alphabetical Paragraph List	Paragraph		Sort	
LOM	LOM	List of Markers	Markers		Collate	<\$markertext>
AML	AML	Alphabetical Marker List	Markers		Sort	
REF	LR	List of References	References		Collate	<\$referencename>
GENERATED INDEX						
IX	IDX	Standard Index	Marker	Index	Sort	
AIX	IOA	Index of Authors	Marker	Author	Sort	
SIX	IOS	Subject Index	Marker	Subject	Sort	
IOM	IOM	Index of Markers	Marker		Sort	
IREF	IR	Index of References	References		Sort	<\$referencename>

Table 1 Summary of generated file types and contents

Generated file elements

The elements that are to be derived from a book's documents and placed in a generated file are enumerated a list held in the book file. With a book file frontmost, select a generated file. Under the *Edit* menu, an item *Set Up Xxxx* will be active, where Xxxx is Table of Contents, List of Tables, etc. Choosing that item will present a dialog by which elements (paragraph tags or marker types) can be chosen.

Generated file formatting

The formatting of elements book components destined for the Table of Contents is determined by information contained on a reference page named TOC. (During Generate of a TOC, if a like-named reference page isn't present in the file, it will be added.) Formatting for the List of Figures is determined by information contained on a reference page named LOF, and similarly for other elements in Table 1.

The TOC special reference page contains (in a text frame) paragraphs having tags whose names end with TOC. (The tags are ordinarily not catalogued.) For a List of Figures, the LOF special reference page contains comparable paragraphs whose tags end with LOF.

The formatting and contents of these paragraphs are taken as prototypes of the corresponding generated elements. Text content is copied verbatim. Such paragraphs ordinarily also contain *building blocks* that generate variable elements. For example, in a TOC, the building block `<$paratext>` causes the contents of the paragraph text to be included in the TOC; the building block `<$page>` causes the page number containing the paragraph to be included.

If Hypertext links are to be generated, a paragraph tagged ActiveTOC is present; text in this paragraph provides the contents of the hypertext marker. (Formatting of this line is ignored.)

Generate

The *generate* algorithm isn't documented, but it goes something like this. The book's documents are scanned to update numbering, update cross-references, and update text insets as necessary.

Then, generated files are produced. Documents are scanned in book order, and within each document, body page text is scanned in flow order. (Often, each document will have just one flow.) Depending upon the derived file type, paragraphs or markers are scanned.

For each generated file (TOC, LOF, LOT, etc.), any previously generated content is (conceptually) deleted by deleting the first run of paragraphs whose tags end in the derived type. That location is where newly generated content will be placed.

For each qualifying element – that is, for each paragraph that is included in the Derived Tags list, or each marker whose type is included in the Derived Markers list – the appropriate Suffix reference page is consulted, the flow whose name corresponds to the derive type is located, and the first paragraph within that flow whose tag matches the derived type is located. Call that paragraph the prototype paragraph. That paragraph, with its formatting, is effectively copied into the generated file. Any backslash constructs within that paragraph are replaced by the corresponding text character. Then, every relevant building block (such as `<$paratext>`, `<$page>`, or

On special reference page paragraphs, backslash has the special meaning that it has within the Find/Replace dialog or within marker text. For example, `\t` generates a tab character. A literal tab can be used instead, but is harder to see.

In any of the steps described here, if the corresponding special reference page, flow, or paragraph is missing, one will be inserted. One way to make a generated file from scratch is to first allow Frame to generate default elements with default formatting, then to manually reformat these elements.

<\$markertext>) is replaced by the corresponding element from the source file. Such replacement retains the formatting of the prototype; except that any character formatting applied in the source will be carried over to the generated file. Overrides will not be carried over.

Finally, if the derived type involves sorting – for example, a standard index – the generated elements are sorted.

I hope that you found this description to be helpful. I do not claim that it is authoritative or complete. If you have corrections, clarifications, or suggestions, please let me know. ■■■