

DAN EMORY & ASSOCIATES

**A New Approach to
Single-Sourcing**

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

Publication Data

Author: Dan Emory

Email: danemory@globalcrossing.net

Publication Date: June, 2003

Version: 1

Copyright Notice

Copyright 2003, Dan Emory & Associates. You are authorized to distribute this entire document in PDF format as long as it is left entirely intact, including the business card page at the end. You may also cite excerpts from this document, provided attribution is given to the author.

About This Document

This structured document was created in a single FrameMaker file, using the ProcBook EDD.

Abstract

This document describes a new approach to single-sourcing with structured FrameMaker documents, using many of the unique features of structured authoring.

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

CHAPTER 1: Introduction

1-1 What is Single-Sourcing?

Single-sourcing utilizes a single base set of “master” FrameMaker documents to accomplish any or all of the following goals:

1. Produce Different Document Versions for Different Product Configurations
In the most extreme case, the product configuration differs for each customer. In less extreme cases, you have two or more model variations of the same basic product, each identified by a model number.
2. Produce Different Document Versions for Different Types of Users
Different types of users require different types of information. A supervisor, for example, may only want an overview of the product, and has no interest in the procedural details.
3. Update Document Information from a Product Database
Information in this category, which can most reliably be auto-updated from a product database, might include part numbers, model numbers, and module/component names when such information varies for different product configurations. But auto-updating of such information might also be desirable simply to assure that the documentation reflects the most current information in the product database at document production time.
4. Produce Multiple Types of Deliverables for each version, including:
 - A. Printed Documents
 - B. PDF documents
 - C. HTML, XHTML, or XML documents
 - D. On-line help in a variety of formats.

In each of the types of deliverables listed in items 1, 2, and 4, there are usually significant differences in content.

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

1-2 Limitations of FrameMaker

FrameMaker provides the following features which are useful in developing a single-sourcing solution, but these features have limitations, particularly in complex documents.

1. Conditional Text

Text strings within paragraphs, entire paragraphs and even graphics and tables can be marked as conditional text.

By hiding all inapplicable text in a master document, it can be configured for a particular type of document deliverable, and/or a particular product configuration. However, when conditional text must be used for both of those purposes (i.e., a combination of a particular document deliverable type and a particular product configuration), the conditional text feature is often not robust enough to effectively handle such situations.

Moreover, as the complexity of the conditional text settings increases, the probability of human mistakes in applying the correct conditional text settings and making the correct show/hide selections for a particular document version increases exponentially. The problem with conditional text becomes particularly acute when there is overlap among two or more competing conditional text settings.

2. User-Defined Variables

These variables are often used for model numbers, part numbers, product names, customer names, and similar information that varies between different product configurations. For each product configuration, the author must laboriously define the correct value for each such variable in the variable catalog in order to reflect a given product configuration. Once again, there is a high probability of human error when many variables must be changed for each product configuration.

3. Text Insets

When text insets reside in individual text flows in the body pages of separate FrameMaker files, management of text insets, and maintenance of cross-references to text insets, becomes problematic.

The main objective of this paper is to describe methods for overcoming these limitations.

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

1-3 ProcBook is Designed for Single-Sourcing

The fundamental concept of information chunking is the most essential ingredient of effective single-sourcing, and structured document authoring is the optimal way to build documents out of chunks. The ProcBook EDD used to create this document is designed (among other things) to facilitate single-sourcing. This Section, for example, is characteristic of the optimal structure of typical single-source documents, as described below:

This Section is an Exemplar of the Optimal Structure

The exemplar Section shown on this page consists of a parent Section element (there are 4 levels of sections in ProcBook), which serves as the container for the Section title (at the top of this page), and one or more Chunk elements, each of which can optionally have a subtitle, followed by a “kitchen sink” of document objects, including ordinary text, lists of various type, notes, cautions and warnings, graphics, equations and tables. Within this Section, for instance, the structure is as follows:

1. The Section Title Element at the top of this page
2. A “glue text” Chunk containing the first paragraph below the section title.

As a general rule, “Glue Text” is intended to be applicable to all document deliverables in which the parent Section is not hidden.

3. This Chunk

It includes the chunk subtitle, the text paragraph that follows it, and this list. If this chunk were applicable only to certain document deliverables, it might be a structured text inset, or it might be hidden when it is not applicable.

4. The “References” Chunk below.

By placing all cross-references and/or hypertext links in a separate chunk (not part of any text inset) within a section, all cross-references and/or hypertext links for this section are activatable and verifiable, even if the source is in a text inset. Individual references can be hidden if they are not applicable to a particular deliverable.

REFERENCES: To review what is meant by single-sourcing, see [1-1 What is Single-Sourcing?](#)

To learn more about implementing the optimal approach, see [CHAPTER 2: Features of the Optimal Approach](#)

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

CHAPTER 2: Features of the Optimal Approach

The features of the optimal approach described in this Chapter are all implementable using the ProcBook EDD.

2-1 Multiple Chapters and Appendices in the Same File

When consecutive front-to-back numbering (without a chapter number prefix to the page number) is used, ProcBook allows multiple chapters and appendixes to be contained within a single FrameMaker file.

Numbering of chapters and sections is an option (the option is used in this document). When numbering is used, the numbering is re-started for each chapter. For documents of small or moderate length, this feature can be of considerable advantage in single-sourcing applications.

This FrameMaker file, for instance includes not only multiple chapters, it also includes the document's front matter.

Of course, ProcBook also permits the creation of separate FrameMaker files for each chapter and appendix. In that case, a structured book file is used to produce the deliverables.

2-2 Metadata

In FrameMaker structured documents, metadata (data about data) is provided in the form of attributes. The following ProcBook attributes are of key importance in single-sourcing applications:

1. The Effect Attribute

This Strings-type attribute is used to indicate the *effectivity* of a particular structural element. It allows you to list (in a separate string for each deliverable) those deliverables in which that element (and its content, of course) should be included. No value in this attribute is assumed to mean the element and its content is included in all deliverables.

2. The ShowHide Attribute


This is a choice-type attribute with two values--Show and Hide. Show is the default value. With the Attribute dialog displayed, you can use Find/Change to search for all elements in which the Effect attribute has a value.

When a value in the Effect attribute is found, evaluate it to determine whether that element and its content is applicable to the document deliverable currently being produced. If the element and its content is inapplicable, you use the Attribute dialog to set Show/Hide to Hide.

DAN EMORY & ASSOCIATES


A New Approach to Single-Sourcing

It can be seen from the foregoing that only one conditional text setting (named "Hide") suffices for producing any document deliverable. Having set the ShowHide attribute of non-applicable elements to "Hide", you simply open the Conditional Text dialog and perform a Find/Change for a value of "Hide" in the ShowHide attribute. Each time an element is found whose ShowHide attribute is set to "Hide", the "Hide" conditional text setting is applied to it. Then, you simply hide all conditional text having the "Hide" setting.

NOTE  Using FrameScript or the FDK, it should be easy to automate the entire process described above. The application would take as its input one or more values of the Effect attribute. A second application would restore the master document to its deliverable-neutral condition by restoring all ShowHide attributes to the Show value, and restoring all components to unconditional text.

2-3 Text Insets on Reference Pages

Text insets are created (or imported from a text inset master document having all text insets on reference pages) as structured ProcBook text insets on reference pages within the master document itself.

NOTE  Admittedly, this solution does not work for the case where a text inset can't fit within a single text frame on a single reference page.

How Text Insets on Reference Pages are Imported

The procedure for importing a text inset by reference or copy is the same as if you were importing it from an external file. But instead of selecting an external file, you select the filename of the master document itself, and, in the Import Text Flow dialog, you select the reference page flow name for the text inset to be imported.

Advantages of this Approach

- In effect, text insets on reference pages within a master document serve the same function as a database repository in which each "information chunk" is contained in a separate record. Multiple variations of a document are created by selecting the applicable text inset "chunks" and stringing them together in the proper order.
- All modifications to text insets are accomplished within the master document itself.
- You don't have to worry about preservation of pathnames to separate text inset files when you move a document (or a derivative version of it) to a different folder of platform.

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

- If cross-references under each Section are placed in separate “References” Chunks (as shown in on page 5), and if cross-reference markers are used in text insets to indicate the cross-reference source, the management of cross-references whose source is in a text inset is greatly simplified, because:
 - It is easier (than would be the case if text insets reside in separate FrameMaker files) to assure that each cross-reference marker in each text inset is unique.
 - All internal cross-references are testable. If there is an unresolved internal cross reference within a REFERENCES chunk, it must be because that text inset has been hidden (or not included) in the master document. Consequently, that cross-reference should be hidden within the “References” chunk by applying the “Hide” conditional text setting to it.

Structural Flexibility

Many ProcBook elements are valid at the highest level, permitting any of the following to be used as the highest-level element in a text inset:

1. An entire Chapter or Appendix
Chapters and Appendixes are made up of one or more Sections
2. An entire Section (any of 4 levels)
A Section is made up of a title, one or more Chunk elements, and any lower-level Sections.
3. Chunk
This is the “kitchen sink” element, which can contain any or all of the following:
 - A. A chunk subtitle
 - B. Ordinary text paragraphs (element Para)
 - C. Code (element CodePara)
 - D. Any type of list
 - E. Graphics
 - F. Equations
 - G. Tables
 - H. Notes, Cautions and Warnings
 - I. References (element Fmrefgrp)
4. Any type of list
5. Graphic
6. Table
7. Equation

Text Insets on Reference Pages

Page 8

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

8. Notes, Cautions, and Warnings

It can be seen, therefore, that all of the following are possible:

- An entire Chapter or Appendix that is a text inset.
- A Chapter or Appendix in which some of all of its Sections are text insets.
- An entire Section that is a text inset.
- A Section in which some or all of its constituent Chunks and/or lower-level Sections are text insets.
- An entire Chunk that is a text inset.
- A Chunk container in which some or all of its constituent elements (listed in items 4 thru 8 above) are text insets.

Management of Text Insets

Another opportunity when text insets are on reference pages is that you can specifically identify the applicability of each text inset, even including its correct positioning within the master document. This is accomplished as follows:

1. A separate “placeholder” text frame is placed above each “actual” text inset text frame. The placeholder text contains explicit instructions about where, within each applicable deliverable, the “actual” text inset should be inserted, as well as the deliverables in which it is to be included or excluded.
2. The text instructions within each such “placeholder” are placed within the same type of top-level container element as its “actual” text inset companion, and its text frame has a unique flowname. Thus the placeholder itself can be imported as a text inset which serves as a placeholder within the body of the master document. Then, at production time for a particular deliverable, the inapplicable placeholders are deleted, and the applicable ones are replaced by their associated “actual” text insets.

2-4 Exploiting the Comment Conditional Text Setting

The default color for this setting is red, which is ideal. Use it for placeholders with special instructions relating to each deliverable. Of course, the comments are hidden at production time.

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

2-5 Management of User-Defined FrameMaker Variables

When user-defined variables are employed to handle situations in which information (e.g., a model number, a part number, a name or different running header/footer text) differs in different deliverables, there needs to be a way to easily and unambiguously determine the proper definitions of those variables in each deliverable.


The solution is to create a reference page containing a table which provides the required information about each such variable for each deliverable. A simple example of such a table is shown below:

Table 2-1: Variable Definition Update Table

Variable (Current Value)	Required Value			
	Deliverable A	Deliverable B	Deliverable C	Deliverable D
Model 1310	Model 1310	Model 1311	Model 1312	Model 1313
Controller	Controller			
Compaq	Compaq	Dell	NexCom	Apple
If current value of variable must be changed: (1) Double-click on Current Value in first column above to open Variable dialog. (2) Click Edit Definition button. (3) In Edit Definition dialog, type new value in Definition slot, and click Change button. (4) Click Done button. (5) Click Done button in Variable dialog.				

To make the required changes needed for each deliverable, the person who updates the variable definitions does not even have to know the names of the variables. And, as you can see from the instructions in the table footing, all required changes in user-defined variables are accomplished and verified directly from within the table.

An additional column could be added to the above table to indicate the values for those variables in the deliverable-neutral configuration of the master document. In that column, all variables would have their default values (which could be the names of the variables themselves).

NOTE  The alternative method is to create a separate empty template file for each deliverable, in which the values of the variables are set for the applicable deliverable. The variable definitions are then imported into the master document to configure it for a particular deliverable. However, I believe the method described here is superior, because it is always better to maintain such information within the master document itself. Each time a variable is deleted or added, or its name is changed or its value in a particular deliverable is changed, the author can immediately go directly to the reference page and make the appropriate correction.

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

CHAPTER 3: Database Publishing--An Alternative Approach to Single-Sourcing

UniMerge from Refined Reports (price about \$700) offers an exciting alternative approach to single-sourcing. UniMerge can work with either structured or unstructured FrameMaker documents.

3-1 The Basic Ingredients of a UniMerge Application

The Merging Template In the approach being described here, the merging template is simply the master document saved in MIF format.

The Data Record(s) The data records are contained in an ASCII text file, produced by a database query (or, alternatively, created using Notepad or other ASCII text editor if there is no database source).

Data records can be in character-delimited or fixed-field form, or in the following alternative form:

```
Fieldname1:Value
```

```
|  
|
```

```
FieldnameN:Value
```

Where:

FieldnameN is the name of the record field

Value is the value of that field.

Data records can also contain repeating groups of subrecords (typically produced by relational joins). Those repeating groups can be nested to an unlimited number of levels.

The UniMerge Command UniMerge executes from the MS-DOS or Unix command line. In its simplest form, the command has the following form:

```
UM MergingTemplate DataRecord
```

Where:

UM is the name of the UniMerge executable

MergingTemplate is the stemname of the merging template (in MIF format).

DataRecord is the filename containing the data record(s) to be merged with the merging template.

Suppose, for example, you wanted to produce a UniMerge output for four different deliverables from the same merging template. In that case, the data record file would contain four records--one for each deliverable. UniMerge produces a separate FrameMaker output file for each record.

The Basic Ingredients of a UniMerge Application

Page 11

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

3-2 A Simple Example

The Data Record

The data record has the following fields and values:

Config:Deliverable B

Model:Model 1311

Name:Controller

Cust:Dell

Observe that these four parameters are identical to those in the third column of [Table 2-1: Variable Definition Update Table](#).

The Merging Template

Below is shown a small snippet from a FrameMaker master document which is used to produce four different deliverables--A, B, C and D. The snippet describes a particular component (the Controller) that is present in Deliverables A, B, C, and D.

Each version of the controller requires a different block diagram. The block diagram contained in the merging template is for the Model 1310 Controller, which is used for Deliverable A. For deliverables B, C, and D, however, the graphic must be replaced by the applicable graphic, using the UniMerge MAPFILE command.

Unimerge commands are shown in the Courier font. Comments are preceded by “//”

^Model ^Name for ^Cust3

Shown below in Figure 3-1 is a block diagram of the ^Name:

```
IF (Config = "Deliverable A")//KEEP EXISTING GRAPHIC
FILE BD1310
ELSE IF (Config = "Deliverable B")//REPLACE GRAPHIC
FILE BD1310 WITH BD1311
    MAPFILE "BD1310" "BD1311"
ELSE IF (Config = "Deliverable C")//REPLACE GRAPHIC
FILE BD1310 WITH BD1312
    MAPFILE "BD1310" "BD1312"
ELSE IF (Config = "Deliverable D")//REPLACE GRAPHIC FILE
BD1310 WITH BD1313
    MAPFILE "BD1310" "BD1313"
END
```

BLOCK DIAGRAM OF MODEL 1310 CONTROLLER

Figure 3-1: Block Diagram of ^Name

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

Shown below is the resulting output that would be produced by UniMerge for the Data Record that was merged. Observe that the block diagram which was contained in the merging template has been replaced by the applicable block diagram for Deliverable B.

=====

Controller Model 1311 for Dell

Shown below in Figure 3-1 is a block diagram of the Controller.



Figure 3-1: Block Diagram of Controller

=====

3-3 Additional Information About UniMerge

UniMerge Syntax and Commands

- You can insert a fieldname anywhere within text by using **^fieldname**, where fieldname is the declared name of a record field. At merge time, Unimerge replaces ^fieldname with the field value.
- UniMerge recognizes a command by virtue of the command's paragraph format tag. All paragraphs having the tag **umcmd** are recognized as commands to be executed by UniMerge at merge time. All such commands disappear from the UniMerge output.
- In addition to placing commands in whole paragraphs, you can insert commands in-line within ordinary text paragraphs. The example below (shown in quotes) demonstrates how this can be used:
"There ^ [IF (num = 0)] is no way ^ [ELSE IF (num = 1)] is only one way ^ [ELSE IF (num = 2)] are two ways ^ [ELSE] are many ways ^ [END] to skin a cat."
In the above example, if the value in the "num" field were 1, UniMerge executes the in-line command at merge time, producing producing "There is only one way to skin a cat".
- In addition to the **IF-ELSE** and **MAPFILE** commands used in the simple example above, UniMerge provides a number of other commands including:
 - Commands for processing repeating groups of subrecords (typically produced by relational joins) within the main record.
 - An **EXEC** command that executes an operating system command while merging a document.

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

- `FIELD` and `FORMAT` commands for declaring and formatting record fields
- `FRAGMENT` and `INCLUDE` Commands for inserting tagged text fragments into the output.
- A `SET` command used to create and/or set the value of report and system variables.

More About IF-ELSE Constructs

- If an `IF` or `ELSE IF` statement evaluates to `TRUE` because the specified field value is present in the current data record, then the text or table or graphic (or another UniMerge command) that appears below that statement is included in the output or executed. For example:

```
IF (config = "Deliverable A"
```

```
Text for deliverable A.
```

```
ELSE IF (config = "Deliverable B"
```

```
Text for deliverable B.
```

```
END
```

There are only three possible outcomes, as follows:

1. The `IF` statement is true, in which case "Text for deliverable A" is included in the output.
 2. The `ELSE IF` statement is true, in which "Text for deliverable B" is included in the output.
 3. Neither statement is true, which means that, for other deliverables (e.g., C and D), nothing is output.
- In addition to the `=` operator, the operators `!=` (not equal), `>` (greater than), `<` (less than), `>=` (greater than or equal to), `<=` (less than or equal to) and `NOT` can be used in `IF-ELSE` expressions.

- Expressions can be connected by `OR` and `AND`. For example:

```
IF (fieldA = "X" AND fieldB = "Z")  
OR (fieldA = "Y" AND fieldB = "W")
```

- Expressions can be nested to any number of levels. For example:

```
IF (fieldA = "X")  
    IF (fieldB = "Z"
```

```
Put in this text
```

```
ELSE
```

```
Put in some different text
```

```
END
```

```
END
```

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

CHAPTER 4: Putting It All Together

At least some of the methods and tools described in Chapters 2 and 3 can enhance almost any single-sourcing project.

And certainly, the concept of information chunking is vital to any effective solution. The best way to formalize a chunking approach is through structured document authoring, using an EDD (such as ProcBook) with highly flexible and adaptable structure. A structured approach has the added advantage of providing metadata which facilitates single-sourcing.

Beyond a relatively low level of complexity, attempting to use conditional text as the sole method for solving all single-sourcing problems with multiple complex conditional text settings is a guaranteed management nightmare. As described in [2-2 Metadata and Management of Text Insets](#), it is possible to greatly simplify, and more effectively manage, the use of conditional text, even in complex projects.

Similarly, the use of large numbers of user-defined variables to handle variations in content among different deliverables and products becomes unmanageable and error-prone without some sort of management overlay. A solution is offered in [2-5 Management of User-Defined FrameMaker Variables](#).

A solution to the use and management of text insets, including solving problems with cross-references to text insets, is provided in [2-3 Text Insets on Reference Pages](#).

Finally, the radical new solution described in [CHAPTER 3: Database Publishing--An Alternative Approach to Single-Sourcing](#) can, in its ultimate form, eliminate most (or even all) requirements for using conditional text and user-defined variables, as well as providing a means to update information from a product database source.

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

4-1 There are Three Basic Approaches to Single-Sourcing

The three approaches described below are at the extremes. In most cases, a hybrid of these approaches is optimal.

4-1.1 Skeleton Documents

The document contains only those components which are common to all deliverables. Placeholders (see [Management of Text Insets](#) and [2-4 Exploiting the Comment Conditional Text Setting](#) for details) are inserted for each applicable text inset in each deliverable. At production time for a particular deliverable, the applicable text insets are inserted below their associated placeholders, and all placeholders are then hidden. User-defined variables must also be set to their correct values for the deliverable being produced.

4-1.2 “Kitchen Sink” Documents

Everything (including text insets) for all deliverables is included. At production time for a particular deliverable, metadata (see [2-2 Metadata](#)) is used to determine what material should be hidden (by applying the Hide conditional text setting). User-defined variables must also be set to their correct values for the deliverable being produced.

4-1.3 The Database Publishing Approach

This is a variation on the “Kitchen Sink” approach. UniMerge commands embedded in the document determine which information is to be included in each deliverable. In effect, the UniMerge commands serve the same function as conditional text, and some record field values may serve the same function as user-defined variables, possibly including auto-updating some information with the most current data in the product database.

See [CHAPTER 3: Database Publishing--An Alternative Approach to Single-Sourcing](#) for more details.

4-2 The Compelling Advantages of Database Publishing

The fatal phrase used to describe the process in the “Skeleton” and “Kitchen Sink” approaches is “at production time for a particular deliverable.” It takes considerable time using skilled labor to configure a complex master document for the production of a particular deliverable. Each time this process is performed, there is a strong probability that mistakes will be made, some of which may even corrupt the master document itself.

There are Three Basic Approaches to Single-Sourcing

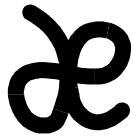
Page 16

DAN EMORY & ASSOCIATES

A New Approach to Single-Sourcing

What's more, during the development of master single-source documents, it is frequently necessary to output any or all deliverables on short notice so that in-process reviews can be performed.

The database publishing approach provides true on-demand publishing. In a matter of a few minutes at most, with no advance notice, any or all deliverables can be produced by UniMerge from the master document, and then opened in FrameMaker for printing or conversion to some other format (e.g., PDF, XML, HTML).

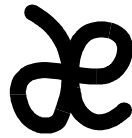


Dan Emory & Associates

Information Design Specialists

Dan Emory

177 Riverside Ave. STE F #1151
Newport Beach, CA 92663
Voice/Fax: 949-722-8971
Email: danemory@primenet.com



Dan Emory & Associates

Information Design Specialists

Dan Emory

177 Riverside Ave. STE F #1151
Newport Beach, CA 92663
Voice/Fax: 949-722-8971
Email: danemory@primenet.com



Dan Emory & Associates

Information Design Specialists

Dan Emory

177 Riverside Ave. STE F #1151
Newport Beach, CA 92663
Voice/Fax: 949-722-8971
Email: danemory@primenet.com



Dan Emory & Associates

Information Design Specialists

Dan Emory

177 Riverside Ave. STE F #1151
Newport Beach, CA 92663
Voice/Fax: 949-722-8971
Email: danemory@primenet.com



Dan Emory & Associates

Information Design Specialists

Dan Emory

177 Riverside Ave. STE F #1151
Newport Beach, CA 92663
Voice/Fax: 949-722-8971
Email: danemory@primenet.com



Dan Emory & Associates

Information Design Specialists

Dan Emory

177 Riverside Ave. STE F #1151
Newport Beach, CA 92663
Voice/Fax: 949-722-8971
Email: danemory@primenet.com



Dan Emory & Associates

Information Design Specialists

Dan Emory

177 Riverside Ave. STE F #1151
Newport Beach, CA 92663
Voice/Fax: 949-722-8971
Email: danemory@primenet.com



Dan Emory & Associates

Information Design Specialists

Dan Emory

177 Riverside Ave. STE F #1151
Newport Beach, CA 92663
Voice/Fax: 949-722-8971
Email: danemory@primenet.com