FOSI *Quickstart* Tutorials

for Arbortext Editor™

Part 1: Formatting for Screen and Print FOSIs

Suzanne Napoleon
www.FOSIexpert.com

This workbook was authored using Arbortext Editor with the EZ interface and a DocBook-based custom DTD. It was formatted for print with a custom FOSI.

Version 1.0

Contents

PREFACE

The FOSI Quickstart tutorials take a learn-by-doing approach to teaching the necessities of the FOSI formatting language for Arbortext Editor, while creating useful code you can refer to and reuse later. The tutorials also teach you how to make the most of Arbortext's interfaces for FOSI development and maintenance.

Audience

The tutorials are intended for: those who want to learn the fastest, simplest way to develop a FOSI that is as easy as possible to maintain; those who want to learn how to utilize the style panels interface; and those who would like to refresh their knowledge of FOSI. The tutorials are also useful for those who are interested in learning what FOSI can contribute to an Arbortext Editor application.

Prerequisites

A working knowledge of structured markup, Arbortext Editor, and DTDs is a prerequisite for learning FOSI formatting. Programming experience, however, is not required. The most useful background is desktop publishing, typesetting, or other document preparation experience.

What’s in this workbook

This workbook contains Part 1 of the FOSI Quickstart Tutorials, which covers the FOSI formatting capabilities that apply to the Arbortext Edit window display. This formatting is a subset of the formatting that applies to print/PDF output with Arbortext Print Composer or Arbortext Publishing Engine. Formatting just for print/PDF output is covered in the Part 2 tutorials. For more information on Part 2 tutorials, please refer to **What’s not in this workbook** below.

An overview of FOSI formatting precedes the tutorials. A chapter with useful information for FOSI development follows the tutorials. The last chapter outlines the author’s methodology and best practices for FOSI development. The first appendix contains the OutSpec DTD supported by Arbortext with annotations indicating what is supported for Edit window display. The second appendix is a short list of other resources related to FOSI.

In addition, this workbook is studded with margin notes containing tips and tricks related to FOSI development and Arbortext applications.

What’s not in this workbook

This workbook does not contain Part 2 tutorials for FOSI formatting that applies only to print/PDF output (and has no effect on the Edit window display). Part 2 tutorials will be available separately.

Conventions used in this workbook

For brevity, this workbook uses “formatter” or “formatting engine” rather than Arbortext Print Composer and Arbortext Publishing Engine.

FOSI categories and characteristics are shown with initial caps to highlight them. SGML/XML element names have start (<) and end (>) tags. Attributes referred to in text are formatted in italic.

CHAPTER 2

FOSI Quickstart Tutorials Part 1

The Quickstart tutorials in this workbook cover formatting that applies to the Arbortext Edit window display and to print/PDF output. Those who need a FOSI for print/PDF output also need to take the tutorials in Part 2, which will be available separately. Part 2 tutorials cover FOSI components related just to print/PDF output, so if you just need to develop a screen FOSI, you do not need to take Part 2.

These tutorials demonstrate how to code as little as possible to achieve the desired formatting while making the FOSI as easy as possible to revise in the future. The idea is to specify formatting once so a single change will affect all relevant elements. This means maximizing “global” formatting and minimizing “local” formatting.

Arbortext Editor has two interfaces for FOSI development and maintenance: the style panels interface and the tagged editor. Each has its strengths and weaknesses, so it makes sense to use both. The style panel interface uses more familiar formatting terms than the tagged editor. The tagged editor displays the FOSI as a linear document, while the style panels interface lets you be

FOSI TIP

End users dislike seeing error messages. For FOSI development, you need to see all warnings, including all errors, warnings, and informational messages for generated text, so you can fix the problem and eliminate the messages. You can select **Tools→Preferences→Warnings** in the Edit window to set these preferences, or you can enter **`set gentextwarnings=3`**, **`set formatwarnings=on`**, and **`set fosiwarnings=on`** at the command line, or you can put these commands in an ACL file and source it before working on a FOSI.

in more than one place in the FOSI at the same time. All but the last two tutorials in Part 1 use the style panels interface. In Part 2, both interfaces are used.

All tutorials use the same DTD, which is designed for training purposes only and is not intended as a production DTD. Select **Tools→Document Type Viewer** in the Edit window to display the DTD Viewer and explore the DTD. DTD elements and attributes are named for their purpose and are spelled out whenever feasible rather than abbreviated in order to make their meaning clear. Not all elements and attributes, however, are used in Part 1 tutorials.

Each tutorial has its own XML (.xml) file and associated FOSI (.fos) file. A copy of each FOSI with all the coding from that tutorial is also included, so you can compare results. To use the completed FOSI, in the Arbortext Edit window, select **Format→Select Stylesheets...**

NOTE: Be sure to take the tutorials in order. Each tutorial builds on things you learned in previous tutorials. Some tutorials are followed by a short **Q and A** section to provide some food for thought and things to try.

NOTE: Some tutorials deliberately create common error conditions so you can learn how to fix them. Consequently, it is important that you perform each step exactly as stated and in the designated order. There's a check box at the end of each step so you can keep track of which steps you've completed. After successfully completing a tutorial, go ahead and experiment on your own before closing the document.

JARGON

Character fill means one character is repeated to fill the available horizontal space, such as the dots in a table of contents between the text and the page number.

JARGON

Charfill is the OutSpec term for character fill.

FOSI TIP

Common character fills are dots, spaces, and dashes. However, any character can be repeated, including a character entity such as **`&mdash;`**.

JARGON

The OutSpec calls a text variable with **Forward Reference Status=yes** a **time-independent variable (tiv)**. **doc-title.txt** in **Tutorial 9** is a time-dependent variable (tdv), which means its value is immediately available without any extra formatting passes.

FOSI TIP

The text variable's **.app** extension indicates it is an appended variable.

FOSI TIP

To support "Sheet n of m" for a <figure> element with multiple pages, use a time-independent variable scoped to <figure>.

The next tutorial introduces character fills, further explores text variables, and creates a table of contents.

TUTORIAL 1-11: Code a table of contents

New: character fill, time-independent variable, generated file

1. Open `tutorial1-11.xml`. This FOSI builds on the previous FOSI.
2. Start the All Character Fills panel.
3. Click on "All Character Fills." Then select **Edit→Add**, or use the right mouse button to select **Add charfill**, or use the **Ctrl+a** keymapping.
4. With the cursor in the field after "Character Fill," enter **`dotfill`**, then **TAB** to the field after "String to Repeat" and enter a period: **.**
5. Close the All Character Fills panel.
6. Start the All Text Variables panel to add a new text variable for storing <title> elements for the table of contents. The variable will be output from a Use Text Source in the <toc> e-i-c.
7. Add a text variable named **toc.app** with **Forward Ref. Status=yes**, **Scope Is Local to Element=document**, and **Link Use and Save Locations=yes**.

   NOTE: Setting **Forward Reference Status=yes** tells the formatter that **toc.app** is empty when it is first encountered in the document because its contents come from elements later on in the document. The formatter makes at least one more formatting pass through the document in order to output the contents of the variable.

   Scope Is Local To Element=document tells the formatter the elements that save to the variable are all children of the <document> element. In other words, the table of contents applies to the entire document. For a Table of Contents for each <chapter>, for example, code **Scope Is Local to Element=chapter**.

continued...