

# CHAPTER 4

## FOSI Variables

---

FOSI has two kinds of variables: **counters** and **strings**. Their case-insensitive names are IDs.

As discussed in the following pages, counter variables are used to generate numbering (or lettering) for pages, divisions, list items, and the like. String variables are used to store the value of element content, counters, other strings, and other FOSI constructs. In addition, the value of a counter or string can be tested and used as a FOSI characteristic.

Within the page description, counters and strings are processed in the following order:

1. page resources
2. page header
3. page footer

In other words, as would be expected, the page number for the current page is determined before the header and footer are processed.

The FOSI standard (B.4.3.1.5) describes the order of processing within the style description as follows:

### **FOSI Tip**

Best practice is to name counters for what they count; for example, `chapterct`, `itemct`, `footnotect`. Name strings according to the element and, if relevant, context whose content is saved; for example, `pubno.txt`, `chapter-title.txt`, `part.txt`. Also see **Naming conventions** on page 715.

In general, the processing of the flowing text contents and the `e-i-cs` occurs in a logically continuous fashion while the processing of the `pagedesc` specification that “cuts up” the result into pages happens in an “asynchronous” fashion. Therefore, to allow the `pagedesc` specification to refer to values of FOSI variables that are modified by `e-i-c`'s in the flowing text processing, there must be a mechanism to help synchronize the two disjoint processes. The `TO`, `BO`, `FI` modifiers described in B.3.4.10.9 provide this needed synchronization.

In fact, these modifiers apply only to string variables, not counters. The modifiers are detailed beginning on page 463.

First, more information about counters and strings follows.

## Counter variables

A **counter** is a numeric variable whose value may be zero or a positive integer. A counter must be declared in the FOSI with a `counter` category (page 214).

A declared counter has:

- a unique name
- an optional initial value (the default is the empty string)
- a numbering/lettering style, which can be overridden with a counter modifier (see page 216)
- an optional lettering sequence style for letters beyond “z” and “Z”: `aa`, `ab`, `ac`, ... (the default) or `aa`, `bb`, `cc`, ...
- an optional padding length for leading zeroes with Arabic numbers

The available numbering styles for counter variables are:

- Arabic number (the default): 0, 1, 2, ...
- Lowercase alphabet character: `a`, `b`, `c`, ...
- Lowercase Roman numeral: `i`, `ii`, `iii`, ...
- Uppercase alphabet character: `A`, `B`, `C`, ...
- Uppercase Roman numeral: `I`, `II`, `III`, ...
- Korean characters (all 14 consonants and 10 vowels available for sequence)
- Korean characters (14 consonants only)

## Counter examples

Several counters are declared in the first example. They are used in the following examples as well. Notice that when `transactionct` is output, it is padded with as many zeros as needed to make it a six-digit number.

**Figure 34 Counter declarations**

```
<counter enumid="appenddixct" initial="0" style="alphauc">
<counter enumid="chapterct" initial="0" style="arabic">
<counter enumid="foliocr" initial="0" style="arabic">
<counter enumid="front-foliocr" initial="0" style="romanlc">
<counter enumid="itemct" initial="0" style="alphalc" seq="1">
<counter enumid="next-page-blankct" initial="1" style="arabic">
<counter enumid="outline-level-1ct" initial="0" style="romanuc">
<counter enumid="outline-level-2ct" initial="0" style="alphauc">
<counter enumid="outline-level-3ct" initial="0" style="arabic">
<counter enumid="outline-level-4ct" initial="0" style="alphalc"
<counter enumid="outline-level-5ct" initial="0" style="romanlc">
<counter enumid="sheetct" initial="0" style="arabic">
<counter enumid="transactionct" initial="0" style="arabic" padlen="6">
```

In the next example, the chapter number is incremented and saved to a string variable along with generated text and the contents of the `<title>` element.

**Figure 35 Counter incremented and saved**

```
<e-i-c gi="chapter" context="body">
<charlist inherit="1" charsubsetref="division">
<enumerat enumid="chapterct" increm="1">
...
<e-i-c gi="title" context="chapter">
<charlist inherit="1" charsubsetref="title">
<savetext textid="chapter-title.txt"
conrule="\Chapter \,chapterct,\. \,#CONTENT>
...
```

Following is an example of a counter that is incremented and output along with a generated period and space.

**Figure 36 Counter incremented and output**

```
<e-i-c gi="item" context="list">
<charlist inherit="1" charsubsetref="block">
<enumerat enumid="itemct" increm="1">
<usetext placemnt="before" source="itemct,\. \>
...
```

The last example shows how to use a counter to indicate in a page header or footer that the next page number is blank.

**Figure 37** Counters for current page and following blank page

```
<recto>
<pageres>
<enumerat enumid="foliocr" increm="1">
<enumerat enumid="next-page-blankct" increm="1">
...
<footer>
<usetext source="\Page \,foliocr">
...
<rectobb>
<pageres>
<enumerat enumid="foliocr" increm="1">
<enumerat enumid="next-page-blankct" increm="1">
...
<footer>
<usetext source="\Page \,foliocr",\ (Page \,next-page-blankct,\ blank)\ ">
...
```

## String variables

A **string** variable should be declared with `stringdecl` (page 218). A declared string has:

- a unique name
- an optional initial value (the default is the empty string)
- a declaration that the string is time-dependent, which means its contents are immediately available, or time-independent, which means at least one formatting pass is necessary before the string has content (the default is time-dependent); see **Time-dependent versus time-independent variables** on page 220 for details.
- a setting to link where the string is saved to where it is used

A FOSI string variable may contain any of the following:

- element content
- attribute content
- literal text
- non-keyboard characters
- cross references
- spacing specifications

- padding
- string variable contents
- counter values
- DTD elements, with or without attribute settings
- pseudo-elements
- processing instructions

**NOTE:** A string variable may be appended to in order to accumulate material, such as a table of contents.

## String variable examples

**NOTE:** These examples rely on the code in **Counter examples** on page 65

Following are several examples of string variable declarations.

### Figure 38 String variable declarations

```
<stringdecl textid="chapter-title.txt" literal="" hotlink="1">
<stringdecl textid="first-name.txt" literal="">
<stringdecl textid="name.txt" literal="">
<stringdecl textid="note.txt" literal="Nota">
<stringdecl textid="sheetct.txt" literal="">
<stringdecl textid="surname.txt" literal="">
<stringdecl textid="last-sheetct.tiv" literal="" status="1" scope="figure">
<stringdecl textid="names.app" literal="" status="1">
```

In this example, the value assigned when the string is declared (Spanish for “note”) is output as a constant value.

### Figure 39 Output a constant value

```
<e-i-c gi="note">
<charlist inherit="1" charsubsetref="admonition">
<usetext placemnt="before" source="note.txt">
...
```

String variables can be saved to other string variables, as shown in this example.

### Figure 40 Strings saved to a string

```
<savetext textid="name.txt" conrule="first-name.txt, \ \, surname.txt">
```

In the next example, `name.txt` is wrapped with a DTD element (`<para>`) and appended to a string variable to create a list of names.

**Figure 41 String saved to an appended string**

```
<savetext textid="names.app" append="1" conrule="<para>,name.txt,</para>">
```

The last example illustrates how to use a counter and a time-independent string variable to output “Sheet n of m” for each graphic in a multi-graphic figure.

**Figure 42 String variables output “Sheet n of m”**

```
<e-i-c gi="figure">
<charlist inherit="1" charsubsetref="block center">
<savetext placemnt="after" textid="last-sheetct.tif" conrule="sheetct">
...
<e-i-c gi="graphic" context="figure">
<charlist inherit="1" charsubsetref="block">
<enumerat enumid="sheetct" increm="1">
<usetext placemnt="after"
source="\Sheet \,sheetct,\ of \,last-sheetct.tif">
...
```

## Pagedesc and styldesc counters and strings

Arbortext Editor distinguishes between pagedesc variables from other variables and imposes certain restrictions:

- A counter that is reset or incremented in the pageres is a **pagedesc counter**, which cannot be reset or incremented outside the pagedesc except in an e-i-c that forces a new page.
- A string variable saved in the pageres is considered a **pagedesc string**, which cannot be saved to, reset, or nulled out outside the pagedesc.
- For a pagedesc counter to be saved or used outside the pagedesc (for example, in a table of contents or index), it must be saved to a string variable in the pageres and output with a modifier (described in **String modifiers** on page 463).
- A counter that is enumerated in the styldesc is considered a **styldesc counter**, and a string variable saved in the styldesc is considered a **styldesc string**. They cannot be changed in the pagedesc.

**NOTE:** FOSI coding that ignores this distinction does not work and may generate an error message.