

CHAPTER 5

FOSI Syntax and Keywords

FOSI technical details are covered in this chapter, specifically:

- **Units of measurement** on page 70
- **Value types** on page 70
- **ID, IDREF, IDREFS** on page 72
- **Delimiters** on page 74
- **Spacing specifications** on page 74
- **Significant record ends** on page 107
- **Keywords** on page 80

TYPE TIP 

1 pica=12 points
In Arbortext Editor, 1 inch=72.27 points

FOSI TIP 

Explicitly coding pt rather than letting it default can be helpful when you search the FOSI as an ASCII file.

FOSI TIP 

Including the pound sign (#) with RGB values can facilitate searching the FOSI as an ASCII file.

Units of measurement

The table below shows the units of measurement that can be used with FOSI characteristic values. Decimals are allowed. The zero before a decimal point is optional.

Table 5 FOSI units of measurement

Unit	Abbreviation	Examples
centimeters	cm	5cm, 0.65cm, .2cm
inches	in	11in, 8.5in, 0.5in, .25in
millimeters	mm	6mm, 10.1mm, 0.5mm, .1em
points	pt	4pt, 0.5pt, .75pt
picas	pi	20pi, 1.25pi
relative units	em†	2em, 0.5em‡, .33em

†The em is a relative unit of measurement equal to the current font size. When the font size changes, the size of the em changes.

‡0.5em represents an en space

Except for relative units, you can mix units of measurement, with or without a plus sign: 1in+2pi is the same as 1in2pi.

Subtraction is allowed. For example: -1pi-4pt calculates to -16pt, -1pt+4pt calculates to +8pt, and 1in-2pi calculates to 48.27pt.

The default unit of measurement is point (pt).

Value types

The tables in **Chapter 8 FOSI Categories and Characteristics** below that describe characteristics refer to the following types of values:

NOTE: FOSI values are case-insensitive.

NOTE: The types of values listed below do not strictly follow the FOSI standard. Instead, they are designed to communicate as easily as possible Arbortext's support for FOSI.

COLOR	An RGB value or a named color. An RGB value must be hexadecimal digits from 0 through F, optionally preceded by a pound sign (#). Named colors are: aqua, black, blue, brown, gray, gray1, gray2, gray3, gray4, gray5, green, lime, maroon, navy, olive, orange, red, teal, violet, white, and yellow. Note that named colors may not be output at “full strength” For example, when <code>highlight background-color</code> is specified as <code>blue</code> , the color is lighter than when <code>#0000FF</code> is specified. Also, <code>gray</code> and <code>gray 3</code> are the same. <code>gray1</code> is the same as <code>#EEEEEE</code> ; <code>gray2</code> is equivalent to <code>#DDDDDD</code> ; <code>gray3</code> matches <code>#CCCCCC</code> ; <code>gray4</code> is identical to <code>#BBBBBB</code> ; <code>gray 5</code> is equal to <code>#AAAAAA</code> .
DEGREES	A positive integer between 0 (zero) and 360. A value of 361 is the same as 1, 362 is the same as 2, etc.
ID/IDREF	An ID is used to uniquely identify a characteristic or set of characteristics so that it can be referenced by an IDREF. See page 72 for details on ID and IDREF characteristics in a FOSI.
INTEGER	A whole number that is either positive, zero, or negative. A keyboard hyphen character (“-”) before the number designates a negative integer. Some characteristics do not accept negative integers.
NAME TOKEN	A string of no more than 8 characters limited to the following: upper- and lowercase letters (“A” through “Z” and “a” through “z”), Arabic numbers zero through nine (“0” through “9”); a period (.); and/or a hyphen (-).
PERCENTAGE	A positive integer, where 100 indicates 100%.
PLACEMENT	Placement may be “before” or “after.” “Before” means after the start tag, but before the beginning of the element content. “After” means after the element content, but before the end tag.
POINTER	A reference to information contained in an external file. A pointer has the attribute type of “entity” and thus requires an entity declaration in the FOSI declaration subset to identify the external file.
SIZE/DISTANCE	A positive or negative number, which may include up to three decimal points, followed by a two-letter abbreviation for a unit of measurement, as shown in Table 5 FOSI units of measurement on page 70. For example “6pt” means six points, and “6.5pt” means six and one-half points. If a unit of measurement is not included, points are assumed.

STRING	A sequence of one or more characters. Character entities may be used for non-keyboard characters. The syntax for string depends on its context in a FOSI.
TOGGLE	0 (zero) or 1 (one). 0 equates to "no," "off," and "false." 1 means "yes," "on," and "true." In the tagged FOSI editor and the ASCII file, 0 and 1 are used. In the FOSI style panels interface, the choices are "no" and "yes."
URI	Uniform Resource Identifier a string of characters that identifies resources on the web: documents, images, downloadable files, services, electronic mailboxes, and other resources.

Syntax that applies only to specific categories and characteristics is described with the relevant categories, as follows:

- **Context syntax** on page 312
- **Table 55 Leftind syntax** on page 349
- **Table 56 Firstln indent syntax** on page 350
- **Table 57 Rightind syntax** on page 350
- **Table 77 Savetext conrule and usetext source syntax** on page 461

ID, IDREF, IDREFS

In the OutSpec DTD, some characteristics are declared with type ID, IDREF, or IDREFS so that FOSI can utilize SGML's built-in linking mechanism.

Table 6 below shows categories with an ID, IDREF, or IDREFS characteristic.

Table 6 ID, IDREF, or IDREFS characteristics

Category	Attribute by type		
	ID	IDREF	IDREFS
hyphrule	language		
charfill	cfid		
counter	enumid		
stringdecl	textid†		
float		flidref	
floatloc	floatid		
pageset	id		
rectopg versopg blankpg		botfloat†	topfloat
pagespec	pgid		
pageref		pgidref	
flowtext		botfloat†	topfloat
column		botfloat†	topfloat
envdesc	envid		
charsubset	charsubsetid		charsubsetref
charlist		envname	charsubsetref
hyphen		lang	
keeps			floatsout
textbrk		pageid	
float		flidref	
Reset			resetlist†‡
Enumerate		enumid	
Savetext	textid†, xrefidtag†		
Subchars			charsubsetref

†NMTOKEN interpreted as the name of the savetext string variable

‡Arbortext extends the botfloat characteric to allow special float keywords

§NMTOKENS interpreted as a list of counter and string variable names

FOSI Tip 

In the tagged FOSI editor, select Tools→IDs and ID References, or enter `show ids` at the command line, to display a dialog with all the IDs and IDREFs in the FOSI. Click on an ID to link to where the ID is defined or referenced in the FOSI.

Delimiters

Four delimiters (including two Arbortext extensions) are used in `savetext`, `conrule` or `usetext` source characteristics:

- A comma (,) is used to separate entries in a `conrule` or `source`, as shown in the following examples.
- Paired back slash characters (\), which surround literal text and character entities. For example:


```
<usetext placemnt="before" source="\Chapter \,chapterct.txt">
```
- Paired exclamation points (!), which enclose elements that include attribute settings and processing instructions (an Arbortext extension). For example:


```
<savetext conrule="!<entry rowsep="1">!,#CONTENT,!</entry>!" ...>
```

NOTE: When the exclamation point delimiter is used, a `conrule` is allowed to contain only the begin tag or only the end tag — as long as all generated content before and after the element are balanced.
- Paired percent signs (%), which cause the enclosed content to be treated as a literal string that should be passed without modification to the TEX formatter within the Arbortext formatting engine for print/PDF output. For example:


```
<usetext source="%$x^2$" ...>
```

Spacing specifications

A **spacing specification** is horizontal space. The amount of space can be fixed or relative.

A fixed spacing spec inserts the specified amount of space in the writing direction on the current writing line. A negative fixed value moves the current location in the opposite direction from the writing direction.

A relative spacing spec refers to a location on the current line (in the writing direction) using what Arbortext Editor refers to as a **kern-to**. The space in between is called “**padding**.” A kern-to is coded as a spacing spec preceded by an at sign (@).

The effect is similar to a “tab stop.” However, if two “tab stops” are too close together, the content may not fit. If the content of the element is too long, it continues through the next “tab stop.”

More than one relative spacing specification can be included in a `savetext` `conrule` or `usetext` source, as illustrated in **Figure 45 “Tab stops” in Edit window** on page 76. Furthermore, fixed spacing and relative spacing specs can be mixed in a `savetext` `conrule` or `usetext` source, as shown in **Figure 47 Right-aligned hanging numbers** on page 79.

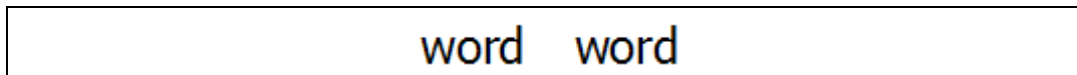
NOTE: Padding works only for the first line of a block element.

Table 5 above shows valid units of measurement for spacing specifications.

Spacing specifications examples

In the first example, a fixed spacing specification separates two words.

Figure 43 Spacing specification in writing direction



FOSI fragment

```
<usetext source="\word\,2pi,\word\"></usetext>
```

When preceded by a minus sign (-), the horizontal space moves the current position in the opposite direction, as illustrated by the following example.

Figure 44 Negative spacing specification



FOSI fragment

```
<usetext source="\&block;&block;\,-16pt,<para.fmt>,\&para;\,</para.fmt>">
<subchars>
<font inherit="1" size="24pt">
</subchars>
</usetext>
...
<e-i-c gi="para.fmt">
<charlist inherit="1">
<font inherit="1" size="0.7em" weight="bold" offset="5pt">
<highlt inherit="1" fontclr="#FFFFFF">
</charlist>
</e-i-c>
```

Relative spacing specification can create “tab stops” in the Edit window. However, as the following figure shows, alignment depends on the content.

Figure 45 “Tab stops” in Edit window

<u>Part</u>	<u>Part Description</u>	<u>Unit Price</u>
12345	A long description about a hammer	\$12.34
67890	A saw	\$56.78
ABCDE	A set of screwdrivers, with a description of each one	\$234.56

XML fragment

```
<partlist>
<part><partno>12345</partno><partdesc>A long description about
a hammer</partdesc><unit-price>12.34</unit-price></part>
<part><partno>67890</partno><partdesc>A saw</partdesc><unit-price>
56.78</unit-price></part>
<part><partno>ABCDE</partno><partdesc>A set of screwdrivers, with a
description of each one</partdesc><unit-price>234.56</unit-price></part>
</partlist>
```

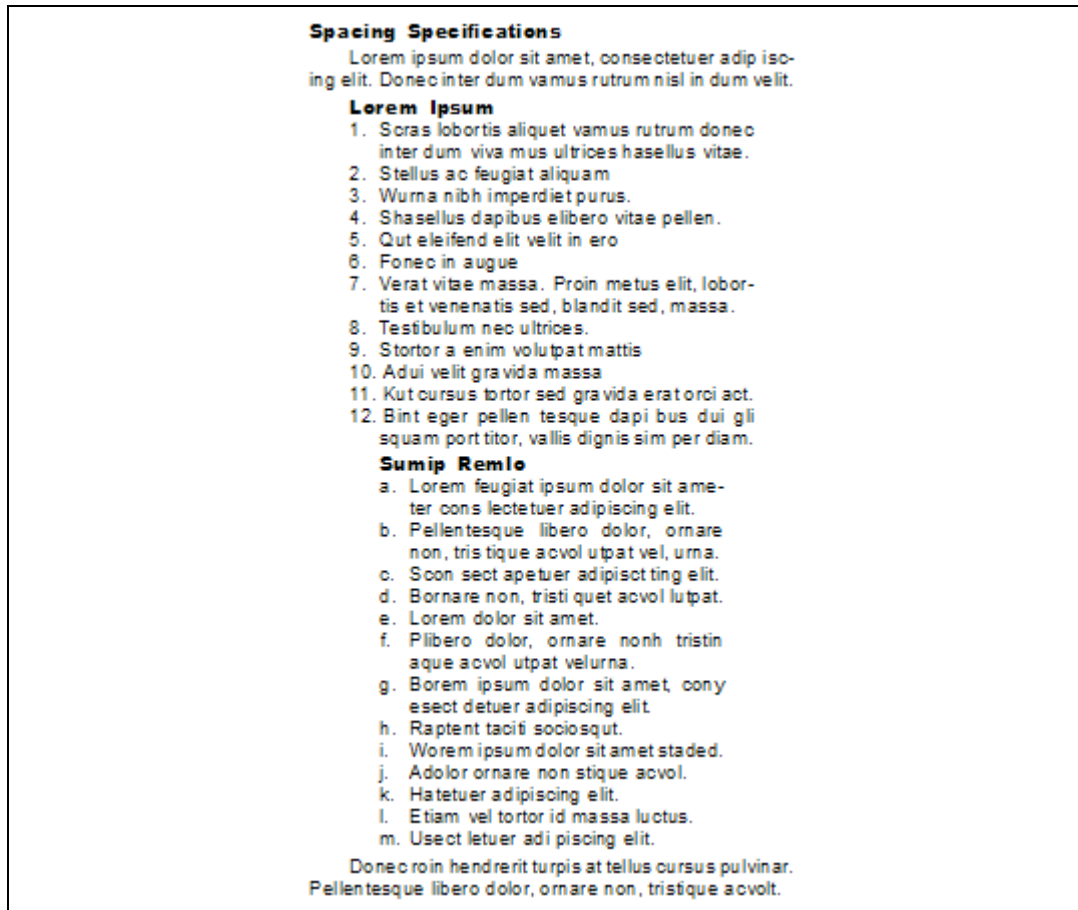
FOSI fragment

```
<e-i-c gi="partlist">
<charlist inherit="1">
<usetext source="@20pi, \ Unit\">
<subchars charsubsetref="startline bold"></subchars>
</usetext>
<usetext source="\Part\, @3pi, \Part Description\, @20pi, \Price\">
<subchars charsubsetref="startline bold underline"></subchars>
</usetext>
...
<e-i-c gi="part" context="partlist">
<charlist inherit="1" charsubsetref="startline">
...
<e-i-c gi="partno" context="part partlist">
<charlist inherit="1">
...
<e-i-c gi="partdesc" context="part partlist">
<charlist inherit="1" charsubsetref="inline">
<usetext source="@3pi"></usetext>
...
<e-i-c gi="unit-price" context="part partlist">
<charlist inherit="1" charsubsetref="endline">
<usetext source="@20pi, \$\"></usetext>
...

```

A relative spacing specification is often used to support list numbering, as shown in the following figures.

Figure 46 Left-aligned hanging numbers



XML fragment

```
<title>Spacing Specifications</title>
<paragraph>Lorem ipsum dolor...</paragraph>
<number-list1><title>Lorem Ipsum</title>
<item1><paragraph>Scras lobortis aliquet...</paragraph></item1>
...
<item1><paragraph>Binteger pellentesque ...</paragraph>
<number-list2><title>Sumip Remlo</title>
<item2><paragraph>Lorem feugiat...</paragraph></item2>
...
```

```

<item2><paragraph>Usect letuer...</paragraph></item2>
</number-list2>
</item1>
</number-list1>

```

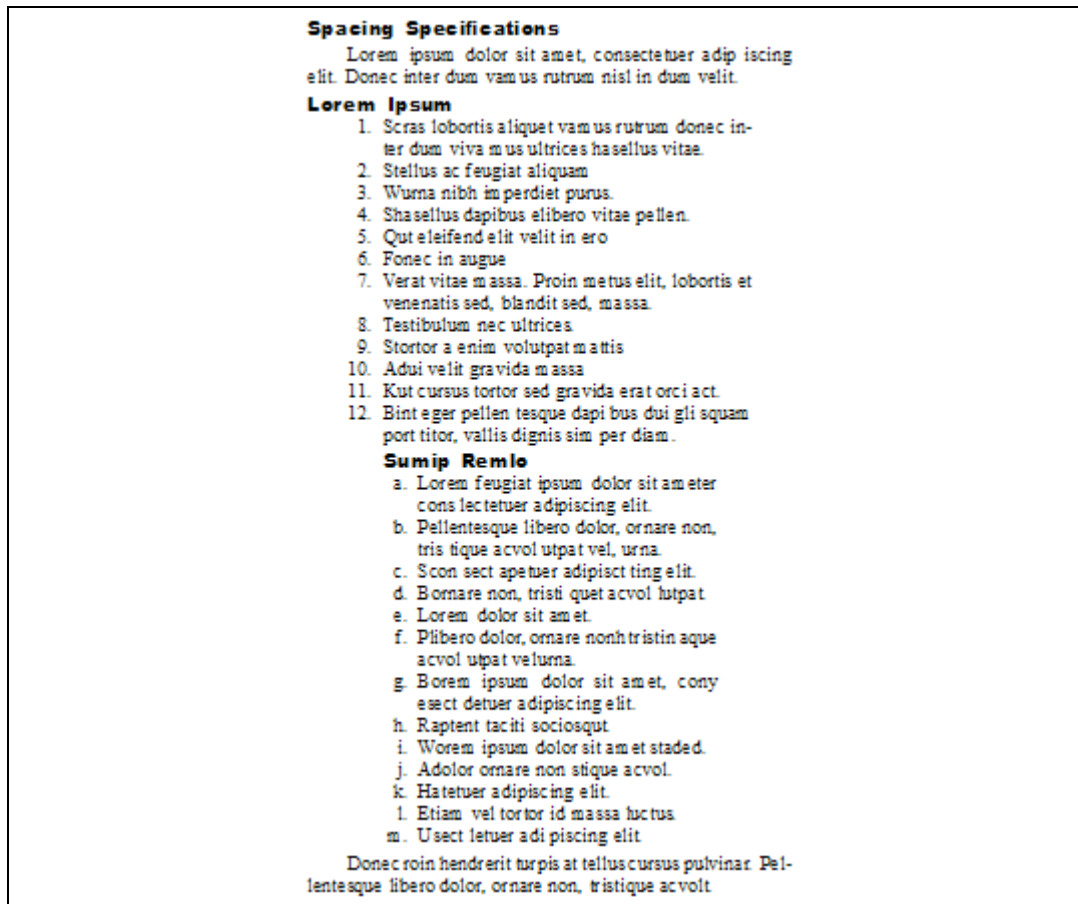
FOSI fragment

```

<counter initial="0" style="arabic" enumid="item1ct">
<counter initial="0" style="alphalc" enumid="item2ct">
...
<e-i-c gi="item1" context="number-list1">
<charlist inherit="1" charsubsetref="block">
<indent inherit="1" leftind="@+1.5em" firstln="*-1.5em">
<enumerat increm="1" enumid="item1ct">
<usetext source="item1ct,\. \,@1.5em" placemnt="before"></usetext>
...
<e-i-c gi="item2" context="number-list2">
<charlist inherit="1" charsubsetref="block">
<enumerat increm="1" enumid="item2ct">
<usetext source="item2ct,\. \,@1.5em" placemnt="before"></usetext>
...
<e-i-c gi="number-list1">
<charlist inherit="1" charsubsetref="block">
<indent inherit="1" leftind="@+2em" rightind="@+2em" firstln="*">
<reset resetlist="item1ct">
...
<e-i-c gi="number-list2" context="* number-list1">
<charlist inherit="1" charsubsetref="block">
<indent inherit="1" leftind="@+1.5em" rightind="@+1.5em" firstln="*-1.5em">
<reset resetlist="item2ct">
...
<e-i-c gi="paragraph" context="item1">
<charlist inherit="1" charsubsetref="endline"></charlist>
...
<e-i-c gi="paragraph" context="item2">
<charlist inherit="1" charsubsetref="endline"></charlist>
...
<e-i-c gi="title" context="number-list1">
<charlist inherit="1" charsubsetref="title"></charlist>
...
<e-i-c gi="title" context="number-list2">
<charlist inherit="1" charsubsetref="title"></charlist>
...

```

Figure 47 Right-aligned hanging numbers

**FOSI fragment**

```
<counter initial="0" style="arabic" enumid="item1ct">
<counter initial="0" style="alphalc" enumid="item2ct">
...
<e-i-c gi="item1" context="number-list1">
<charlist inherit="1" charsubsetref="block">
<indent inherit="1" leftind="@+1.75em">
<enumerat increm="1" enumid="item1ct">
<usetext source="spacefill,item1ct,\\.\\,0.5em,@3.75em" placemnt="before">
...
<e-i-c gi="item2" context="number-list2">
<charlist inherit="1" charsubsetref="block">
<enumerat increm="1" enumid="item2ct">
<usetext source="spacefill,item2ct,\\.\\,0.5em,@1.75em" placemnt="before">
```

```
...
<e-i-c gi="number-list1">
<charlist inherit="1" charsubsetref="block">
<indent inherit="1" leftind="@+2em" rightind="@+2em" firstln="*-2em">
<reset resetlist="item1ct">
...
<e-i-c gi="number-list2" context="* number-list1">
<charlist inherit="1" charsubsetref="block">
<indent inherit="1" leftind="@+1.75em" rightind="@+1.75em"
firstln="*-1.75em">
<reset resetlist="item2ct">
.....
<e-i-c gi="paragraph" context="item1">
<charlist inherit="1" charsubsetref="endline">
...
<e-i-c gi="paragraph" context="item2">
<charlist inherit="1" charsubsetref="endline">
...
<e-i-c gi="title" context="number-list1">
<charlist inherit="1" charsubsetref="title">
...
<e-i-c gi="title" context="number-list2">
<charlist inherit="1" charsubsetref="title">
...
```

Keywords

The OutSpec defines several keywords, and Arbortext Editor supports several more. They are discussed as follows:

- **Generic identifier keyword** on page 80
- **Cross referencing keywords** on page 81
- **Savetext conrule and usetext source keywords** on page 86
- **botfloat and footnote keywords** on page 89
- **Attribute rule keywords** on page 92
- **Sorted, merged, grouped material (indexing) keywords** on page 107

Generic identifier keyword

Arbortext supports #MISSING for an e-i-c **generic identifier**. This e-i-c is used for any element that has no matching e-i-c.

NOTE: Context and occurrence attributes are ignored for gi="#MISSING".

Generic identifier keyword example

The following figure shows a FOSI that contains the minimum necessary for a FOSI to compile.

Figure 48 Minimal FOSI

```
<!DOCTYPE OUTSPEC PUBLIC "-//ArborText//DTD OUTPUT SPEC
MIL-PRF-28001 REV B AMEND 1 19961231//EN">
<outspec>
<?Pub Lcl_label="Adept v8.0">
<rsrccdesc></rsrccdesc>
<pagedesc>
<pageset id="document.page">
<rectopg>
<pagespec>
<topmarg>
<botmarg>
<leftmarg>
<rtmarg>
<header></header>
<footer></footer>
<flowtext>
<column></flowtext>
</pagespec>
</rectopg>
</pageset>
</pagedesc>
<styldesc>
<docdesc>
<charlist></charlist>
</docdesc>
<e-i-c gi="#MISSING">
<charlist></charlist>
</e-i-c>
</styldesc>
</outspec>
```

Cross referencing keywords

The #XREF keyword defined by the OutSpec is discussed below, followed by a section on the ArborText extensions for supporting IDREFS cross referencing.

OutSpec keywords

The FOSI standard provides the #XREF keyword to support a choice of output strings for cross referencing.

`#XREF` is used in a `savetext` conrule in the following form:

```
#XREF(xref-attribute-name, textid-name).
```

When `xref-attribute-name` names a valid attribute whose declared value is `IDREF` or `IDREFS` and whose value matches an `ID` defined elsewhere in the current source document instance, the value of the string variable named by the `textid-name` is returned. The value of the `textid-name` is whatever is saved in the `e-i-c` with the `savetext` associated with the matching `ID`.

NOTE: The `e-i-c` does not necessarily have to be for the element with the `ID`. For example, the `ID` attribute may be on a division tag such as `<chapter>`, but the `savetext` is coded for `<title>` in context of `<chapter>` in order to include the `<title>` content. In such cases, the **xrefidtag** characteristic on `savetext` is coded to specify the element with the `ID`. In this example, `xrefidtag` would be set to `chapter`.

`#XREF(xref-attribute-name, textid-name)` evaluates to the empty string in the following cases:

- When `xref-attribute-name` is not the name of a valid attribute for the current element.
- When `xref-attribute-name` is an attribute of the current element whose declared value is neither `IDREF` nor `IDREFS`.
- When `xref-attribute-name` names an `IDREF` attribute that does not match an `ID` in the same document or whose matching `ID` no value in the document.
- When `xref-attribute-name` names an `IDREFS` attribute that matches no `IDs` in the document and/or none of the matching `IDs` has a value in the document.

In the case of `IDREFS`, the `#XREF` construct returns the string that is a concatenation (separated by a single blank) of each string that would be returned by each individual `IDREF` in the same order as listed in `IDREFS`.

Arbortext keyword extensions

`e-i-c` supports two string variables to output generated text between the cross-references resulting from an `IDREFS` attribute containing multiple values:

- `Idrefsep`, whose value is output after all but the last two generated cross references. The default is:

```
<stringdecl textid="idrefsep" literal="," status="0">
```

- `idreffinalsep`, whose value is output between the last two generated cross references. The default is:

```
<stringdecl textid="idreffinalsep" literal="and " status="0">
```

`idrefsep` and `idreffinalsep` can be redefined if desired using `stringdecl` and/or `savetext`. `savetext` is needed if the necessary syntax is not allowed in `stringdecl` `literal` but is allowed in a `savetext` `conrule`. For example:

```
<stringdecl textid="idrefsep" literal="; " status="0">
...
<savetext textid="idreffinalsep" conrule="<linebreak.fmt>,\and \,
</linebreak.fmt>">
```

NOTE: The names `idrefsep` and `idreffinalsep` cannot be changed.

Cross referencing keyword examples

In this example, `<cross-reference>` does not specify a *type* attribute, so the default value *page-long* is used.

Figure 49 #XREF for multiple cross reference strings

The Bottom Margin Area is the area between the bottom edge of the Footer Area and the bottom edge of the Page Area. It runs the width of the Page Area, as shown in **Figure 2-63 Bottom Margin Area** on page 173.

DTD fragment

```
<!ELEMENT figure (title,(graphic|line-for-line)+) >
<!ATTLIST figure cross-ref-code ID #IMPLIED >
<!ELEMENT cross-reference EMPTY >
<!ATTLIST cross-reference
    matching-cross-ref-code IDREF #REQUIRED
    type (short|long|page-short|page-long) page-long" >
```

XML fragment

```
<paragraph>The Bottom Margin Area is the area between the bottom
edge of the Footer Area and the bottom edge of the Page Area. It
runs the width of the Page Area, as shown in <cross-reference
matching-cross-ref-code="a1234">.</paragraph>
...
<figure cross-ref-code="a1234">
<title>Compare a CDATA attribute to a string variable</title>
...

```

FOSI fragment

```
<stringdecl textid="long-xref.txt" status="0" hotlink="1">
```

```

<stringdecl textid="page-long-xref.txt" status="0" hotlink="1">
<stringdecl textid="page-short-xref.txt" status="0" hotlink="1">
<stringdecl textid="short-xref.txt" scope="0" hotlink="1">
...
<e-i-c gi="cross-reference">
<charlist inherit="1" charsubsetref="inline"></charlist>
<att>
<specval attname="type" attloc="cross-reference" attval="short">
<charsubset>
<usetext source="#XREF (matching-cross-ref-code, short-xref.txt) "></usetext>
</charsubset>
</att>
<att>
<specval attname="type" attloc="cross-reference" attval="long">
<charsubset>
<usetext source="#XREF (matching-cross-ref-code, long-xref.txt) "></usetext>
</charsubset>
</att>
<att>
<specval attname="type" attloc="cross-reference" attval="page-short">
<charsubset>
<usetext source="#XREF (matching-cross-ref-code, page-short-xref.txt) ">
</usetext>
</charsubset>
</att>
<att>
<specval attname="type" attloc="cross-reference" attval="page-long">
<charsubset>
<usetext source="#XREF (matching-cross-ref-code, page-long-xref.txt) ">
</usetext>
</charsubset>
...
<e-i-c gi="title" context="figure">
<charlist>...</charlist>
<att>
<specval attname="cross-ref-code" attloc="figure" attval="#ANY">
<charsubset>
<savetext textid="short-xref.txt" xrefidtag="figure"
conrule="<bold.fmt>, \Figure \, chapterct, \-\, figct, </bold.fmt>">
<savetext textid="long-xref.txt" xrefidtag="figure"
conrule="<bold.fmt>, \Figure \, chapterct, \-\, figct, \ \,
#CONTENT, </bold.fmt>">
<savetext textid="page-short-xref.txt" xrefidtag="figure"
conrule="<bold.fmt>, \Figure \, chapterct, \-\, figct, </bold.fmt>,
\ on page \, folio.txt[BO]">
<savetext textid="page-long-xref.txt" xrefidtag="figure"
conrule="<bold.fmt>, \Figure \, chapterct, \-\, figct, \ \,
#CONTENT, </bold.fmt>, \ on page \, folio.txt[BO]">
...

```

The last example illustrates the use of `idrefsep` and `idreffinalsep`.

Figure 50 IDREFS cross references with page number**1. Lorem Ipsum**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec interdum. Vivamus rutrum nisl in velit. For more information, see Chapter 2, Phasellus Dapibus on page 2-1; Chapter 3, Vestibulum on page 3-1; and Chapter 4, Proin Metus on page 4-1.

DTD fragment

```
<!ELEMENT chapter (title,paragraph*,section*) >
<!ATTLIST chapter cross-ref-code ID #IMPLIED >

<!ELEMENT cross-reference EMPTY >
<!ATTLIST cross-reference
    matching-cross-ref-code IDREFS #REQUIRED >
```

XML fragment

```
<chapter cross-ref-code="abc">
<title>Lorem Ipsum</title>
<paragraph>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Donec interdum. Vivamus rutrum nisl in velit. For more information,
see <cross-reference matching-cross-ref-code="def ghi jkl"/>.</paragraph>
</chapter>
<chapter cross-ref-code="def">
<title>Phasellus Dapibus</title>
...
<chapter cross-ref-code="ghi">
<title>Vestibulum</title>
...
<chapter cross-ref-code="jkl">
<title>Proin Metus</title>
...
```

FOSI fragment

```
<stringdecl textid="idrefsep" literal="; ">
<stringdecl textid="idreffinalsep" literal="; and ">
...
<e-i-c gi="chapter" context="body">
<charlist inherit="1">
<textbrk startpg="recto">
<enumerat increm="1" enumid="chapterct">
<savetext textid="chapterct.txt" conrule="chapterct">
...
<e-i-c gi="cross-reference">
<charlist inherit="1" charsubsetref="inline"></charlist>
<att>
<fillval attname="matching-cross-ref-code" attloc="cross-ref"
fillcat="usetext" fillchar="source">
<charsubset>
```

```
<usetext></usetext>
...
<e-i-c gi="title" context="chapter">
<charlist inherit="1" charsubsetref="title">
<usetext source="chapterct.txt,\. \" placemnt="before">
...
<att>
<fillval attname="cross-ref-code" attloc="chapter"
fillcat="savetext" fillchar="textid">
<charsubset>
<savetext conrule="\Chapter \,chapterct.txt,\. \,#CONTENT,\ on page \,
chapterct.txt,\-\,foliact.txt[BO]">
...

```

Savetext conrule and usetext source keywords

The keywords listed below can be used in a savetext conrule and usetext source.

- **#ELEMNAME** returns the name of current element
- **#CONTENT** returns the content of the current element, including the content of any child elements
- **#CONTENT(attribute-name)** returns the value of the specified attribute on the current element. This evaluates to the empty string when:
 - ▶ the specified attribute is not a valid attribute for the current element.
 - ▶ **attribute-name** references an attribute that has been assigned no value in the source document.
 - ▶ **#CONTENT(attribute-name)** references an attribute declared as **ID**, **IDREF**, or **IDREFS**, in which case the value is treated as **CDATA** and returned as text.
 - ▶ **#CONTENT(attribute-name)** references an attribute of declared value **ENTITY**, in which case it evaluates to a string containing the entity name as the value of the attribute.
 - ▶ **#CONTENT(attribute-name)** references an attribute of declared value **ENTITIES**, in which case it returns a string that is the concatenation (separated by a single blank) of the entity names.
- **#XPath(...)#XPath** returns the string value of the XPath expression indicated with The **e-i-c** being processed is assumed to be the current element.
- **#XPathCONTENT(...)#XPath** returns the content of the element represented by the XPath expression indicated with The element whose **e-i-c** is being processed is the current element. The content returned does not

include start and end tags. When the expression results in a node set with more than one element, a `<_sfe:CollectionItem>` pseudo-element wraps each element content. These pseudo-elements can have `e-i-cs` in the FOSI (see **Figure 53 #XPATHTOC and `<_sfe:CollectionItem>`** on page 88).

- `#XPATHTOC(...)#XPATHTOC`, returns the elements represented by the XPath expression indicated with `...`. The element whose `e-i-c` is being processed is the current element. The return value includes begin and end tags, which can have `e-i-cs` in the FOSI, as shown in **Figure 54 #XPATHTOC returns document tags** on page 89.

Savetext/usetext keyword examples

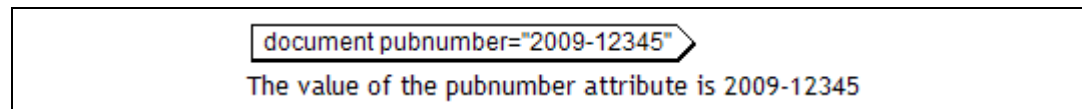
The first example illustrates the use of `#CONTENT` and `#ELEMNAME`.

Figure 51 Output element name and content

```
<usetext source="\The content of element \, #ELEMNAME, \ is \, #CONTENT">
...
```

The next example shows the use of `#CONTENT(attribute-name)`.

Figure 52 Attribute value as content

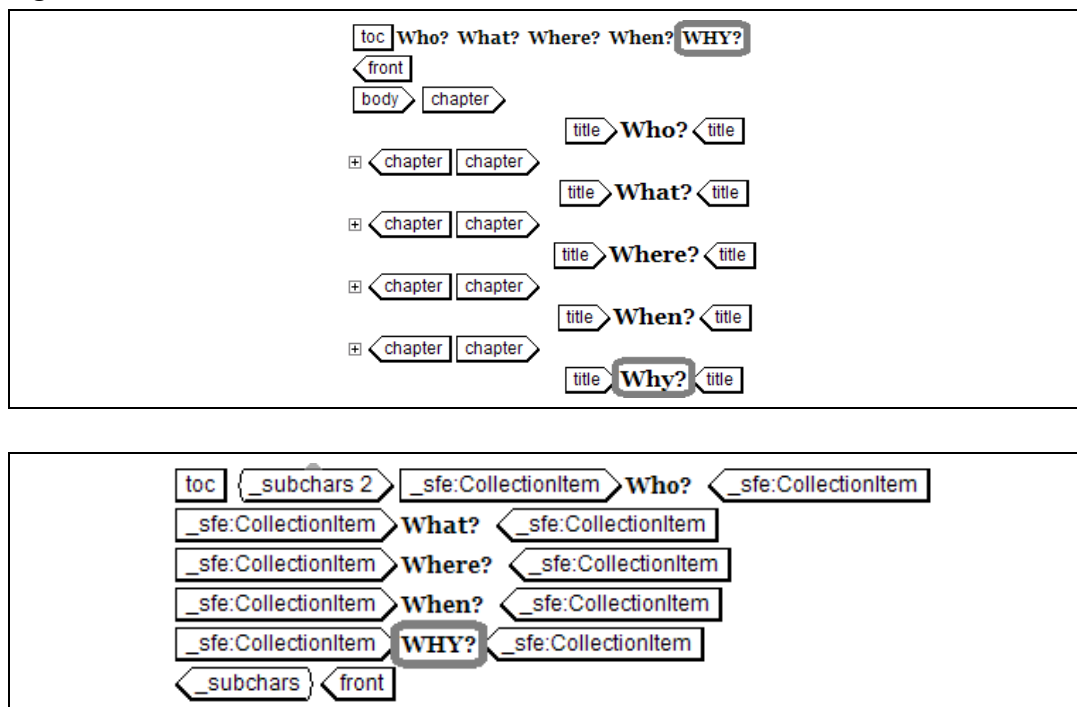


FOSI fragment

```
<e-i-c gi="document">
<charlist inherit="1">
<usetext source="\The value of the pubnumber attribute
is \, #CONTENT(pubnumber) ">
...
```

In the next example, the first graphic shows the `e-i-c` for `<toc>` outputs the value returned by `#XPATHTOC`. In the second graphic, the Edit window has full gentext tags displayed to show the `<_sfe:CollectionItem>` pseudo-element that wraps each `#XPATHTOC` return value when there is more than one. `e-i-cs` can be specified with context and occurrence characteristics to format these pseudo-elements.

Figure 53 #XPATHCONTENT and <_sfe:CollectionItem>

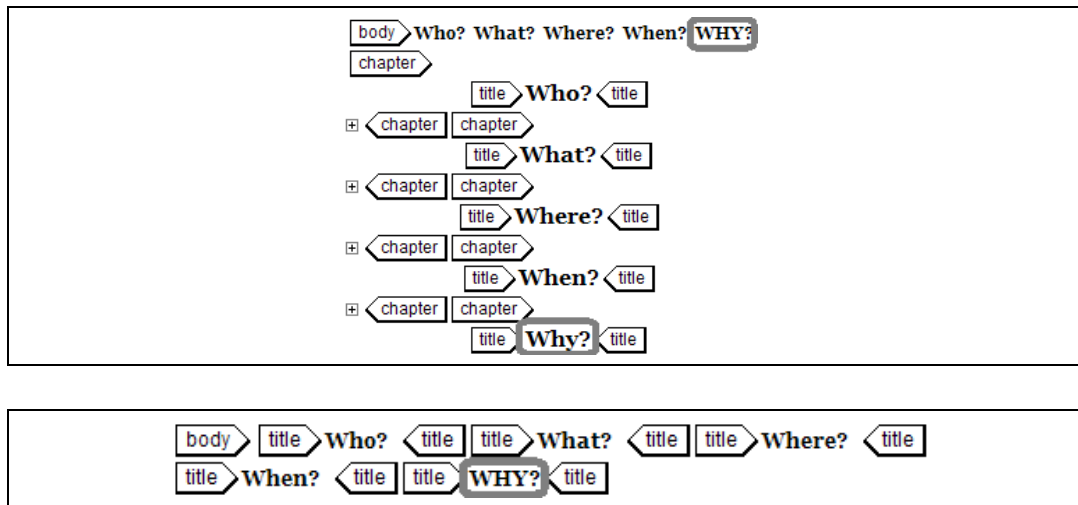
**FOSI fragment**

```

<e-i-c gi="toc">
<charlist inherit="1" charsubsetref="startline">
<usetext source="#XPATHCONTENT(/document/body/chapter/title)#XPATH">
<subchars charsubsetref="bold"></subchars>
...
<e-i-c gi="_sfe:CollectionItem" context="toc" occur="notlast">
<charlist inherit="1">
<usetext source="\ \" placemnt="after"></usetext>
...
<e-i-c gi="_sfe:CollectionItem" context="toc" occur="last">
<charlist inherit="1" charsubsetref="endline allcaps"></charlist>
...

```

In the next example, the first graphic shows that <body> outputs the value returned by #XPATHNODESET. In the second graphic, gentext tags are displayed in the Edit window to show the <title> tag is included. e-i-cs can be specified with context and occurrence characteristics to format these elements.

Figure 54 #XPATHNODESET returns document tags**FOSI fragment**

```

<e-i-c gi="body">
<charlist inherit="1">
<usetext source="#XPATHNODESET (/document/body/chapter/title) #XPATH">
...
<e-i-c gi="title" context="body" occur="notlast">
<charlist inherit="1" charsubsetref="bold">
<usetext source="\ \" placemnt="after"></usetext>
...
<e-i-c gi="title" context="body" occur="last">
<charlist inherit="1" charsubsetref="endline allcaps"></charlist>
...

```

botfloat and footnote keywords

To control footnote placement when a page layout allows floating to page bottom, the following keywords are supported for the botfloat characteristic.

- #Footnote indicates where to put footnotes in relation to other floats. It is allowed once within a botfloat string. It overrides the footnote ftnfloat characteristic, which when set to 1 causes the Footnote Area to appear closest to the flowing text. When set to 0, the Footnote Area appears after any bottom floats. If #footnote does not appear in botfloat, the ftnfloat toggle is honored. If neither is specified, the default is for footnotes to appear before all floats.

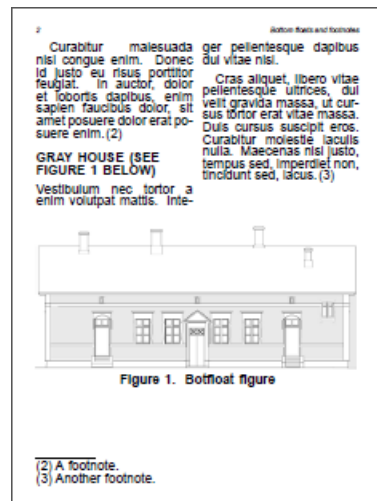
NOTE: #Footnote is not supported in topfloat.

- #Ftnfill indicates the float locations that follow it in the botfloat list are to be bottom-aligned as a group. #Ftnfill is ignored if no floats occur between #ftnfill and #endftnfill. If #Ftnfill is not specified, floatloc minspace, nomspace, and maxspace characteristics are used. For footnotes, minspace, nomspace, and maxspace settings equal to spabove are assumed.
- #Endftnfill is entered in the botfloat list after a #ftnfill to delimit the bottom-justified floats.
- #Ftnsep indicates the location of the footnote separator. If there are no floats after #ftnsep and before #endftnsep, it has no effect. If #ftnsep is not specified, the separator is placed at the top of the Footnote Area.
- #Endftnsep is coded in the botfloat list after a #ftnsep. it restricts the float classes searched for actual floats when deciding if the footnote separator is to appear. If it is not specified but a #ftnsep is present, #endftnsep is assumed to exist at the end of the botfloat list.

Botfloat and footnote example

In this example, the FOSI is coded so the floated figure follows the flowtext. #Footnote positions the footnotes after the bottom float. #ftnfill between them pushes the footnotes to page bottom.

Figure 55 #Ftnfill bottom-aligns footnotes



XML fragment

```

<chapter>
<section>
...
<paragraph>...Nullam dui ... non, metus.</paragraph>
<paragraph>Curabitur ... posuere enim.<footnote>
<paragraph>A footnote.</paragraph></footnote>
</paragraph>
</section>
<section>
<title>Gray House (See <cross-ref matching-cross-ref-code="grayhouse"/>
below)</title>
<paragraph>Vestibulum nec ...</paragraph>
<figure cross-ref-code="grayhouse">
<title>Botfloat figure</title>
<graphic name="gray house.png"/>
</figure>
<paragraph>Cras aliquet ... lacus.<footnote>
<paragraph>Another footnote.</paragraph></footnote>
</paragraph>
</section>
</chapter>

```

FOSI fragment

```

<counter initial="0" style="arabic" enumid="figurect">
<stringdecl textid="figurect.txt" literal="">
<floatloc floatid="figure.bot" floattyp="multiref"
maxdepth="33pi" minspace="2pi" nomspace="2pi" maxspace="2pi">
...
<versopg width="6.5in" nomdepth="8.5in"
botfloat="figure.bot #ftnfill #ftnsep #footnote">
...
<flowtext numcols="2" balance="1">
...
<footnote width="flowtext" ftnsepth="0.5pt" ftnsepln="1in"
spabove="6pt" ftnfloat="0"></footnote>
...
<docdesc>
...
<span span="2">
</docdesc>...
<e-i-c gi="figure">
<charlist inherit="1" charsubsetref="block">
<float flidref="figure.bot" width="page" scope="chapter"
pagetype="afterref">
<enumerat increm="1" enumid="figurect">
<setext source="\Figure \,figurect,\. \,figure-title.txt"
placemnt="after">
<subchars charsubsetref="caption"></subchars>
...

```

Attribute rule keywords

This section discusses keywords supported for `fillval` and/or `specval`, including keywords defined in the FOSI standard and keywords that are Arbortext Editor extensions.

Attloc keywords

Keywords used in the `attloc` characteristic are described below.

#FOSI

[OS B.3.6.5.2.1.b, Tutorial 1–15]

#FOSI is used to test the value of FOSI variables. When `attloc="#FOSI"`, `attname` must contain the name of a declared counter or string variable. The value of the counter or string is used as the value of `attname`.

NOTE: When the counter or string is assigned a value in an `e-i-c`, #FOSI must be coded in a programming pseudo-element rather than directly in the `e-i-c`. Coding #FOSI in the same `e-i-c` causes incorrect results.

For examples of using #FOSI with `specval`, see **Figure 59 Paragraph numbering with optional <title>** on page 101, **Figure 60 Third element outputs horizontal rule** on page 102, and **Figure 259 Screen FOSI displays messages for users** on page 482. On page 628, **Testing if element has content** contains examples that use #FOSI, `specval`, and some “fancy FOSI footwork.” **Figure 65 Fillval with #FOSI** on page 105 illustrates that #FOSI also works with `fillval`.

#XPATH (Arbortext Editor extension)

When #XPATH is coded for `attloc`, the `attname` characteristic is treated as an XPath expression, as demonstrated in **Figure 61 #XPATH with fillval** on page 103, which uses the `fillval` category. **Figure 62 Attloc="#XPATH"** on page 104 shows the use of `specval`.

NOTE: When #XPATH is used with `specval`, `attval` must be 0 (zero) or 1 (one).

The #XPATH keyword is also used with the #XPATHCONTENT and #XPATHNODESET keywords, which are coded with `usetext` and `savetext` characteristics. See **Savetext conrule and usetext source keywords** on page 86 for details and examples.

Attname keywords

Keywords used in the `attname` characteristic are discussed in this section.

SYSTEM-FUNC (Arbortext Editor extension)

[Tutorial 1–15]

When `attloc="SYSTEM-FUNC"`, the `attname` value is treated as the name of an ACL function. An ACL function called by `SYSTEM-FUNC` must be loaded before documents are formatted. The function may be entered at the command line, but for production purposes should be included in a startup file.

If the function exists and has been sourced, it is evaluated and the return value is assigned to the characteristic specified by `fillchar` in the category specified by `fillcat`.

If the function does not exist or has not been not sourced, an error message similar to the following is displayed: “[A30045] Att match: no user defined function named "testfunc". This is a FOSI coding or application ACL error.”

The function must take these arguments:

- `window`, which is a window identifier
- `oid`, which is the oid of the element for the current `e-i-c`

NOTE: Functions used in attribute rules must not modify the source document or errors will occur.

NOTE: Include the ACL package name when calling an ACL function. For example, `opmanual:decode` calls the function `decode` in package `opmanual`.

RELATED ACL

`fosivar_value(window, varname, oid)`. The `fosivar_value` function is called from functions invoked by `attloc="SYSTEM-FUNC"`. Arguments required by the function are:

- `window` is the window ID that is an argument to the function called by `SYSTEM-FUNC`. The value of this parameter can be `-1`.
- `varname` is the name of the FOSI string variable or counter variable whose value is to be returned.
- `oid` is the object identifier (OID) of the element in the user’s document whose `e-i-c` is being processed

FOSI Tip

FOSI code is compiled. However, ACL called by `SYSTEM-FUNC` is executed at run time, and too many `SYSTEM-FUNCS` slow formatting. Best practice is to avoid unnecessary `SYSTEM-FUNCS`.

SYSTEM-VAR (Arbortext Editor extension)

[Tutorial 1–15]

When `attloc="SYSTEM-VAR"`, the `attname` value may be an ACL or environment variable, or a keyword, as described in **Table 7** below.

Table 7 Attname when attloc="SYSTEM-VAR"

Keyword/ variable	Meaning	Examples
<code>editor-only</code>	<p>When <code>attname="editor-only"</code> and <code>attval=#ANY</code>, <code>yes</code>, or <code>1</code>, the attribute rule applies only to the Edit window. When set to <code>#NONE</code>, <code>no</code>, or <code>0</code>, the attribute rule does not apply to the Edit window, but does apply to HTML output from <code>File→Save Aa HTML...</code> and to print/PDF output.</p> <p><code>editor-only</code> is useful for formatting the Edit window to facilitate authoring and editing, including the display of guidelines and warnings.</p>	<p>Figure 64 Editor-only, print-only, html-only on page 104; Figure 74 Optimizing gentext display on page 121; Figure 75 Gentext before and after tag pair in Edit window on page 123; Figure 76 Gentext before and after singleton on page 124; Figure 259 Screen FOSI displays messages for users on page 482</p>
<code>screen-only</code>	Same as <code>editor-only</code> .	See above.
<code>print-only</code>	<p>When <code>print-only</code> is set to <code>#ANY</code>, <code>yes</code>, or <code>1</code>, the attribute rule applies only to print/PDF output. When set to <code>#NONE</code>, <code>no</code> or <code>0</code>, the <code>attspe</code> does not apply to print/PDF output, but does apply to HTML output from <code>File→Save as HTML...</code> and to the Edit window display.</p> <p><code>print-only</code> is useful when print-formatting does not render well for screen display. Also, when <code>suppress sup="1"</code> is coded to apply only to print/PDF output, the suppressed text continues to be displayed in the Edit window even when <code>set hidesuppressed=on</code>.</p>	<p>Figure 64 Editor-only, print-only, html-only on page 104; Figure 75 Gentext before and after tag pair in Edit window on page 123; Figure 197 Refpoint="bottom" on page 364; Figure 219 Inline boxing on page 408; Figure 229 Repeating float on page 430</p>

continued . . .

Keyword/ variable	Meaning	Examples
html-only	When <code>html-only</code> is set to <code>#ANY</code> , <code>yes</code> , or <code>1</code> , the attribute rule applies only to HTML output from File→Save as HTML.... When set to <code>#NONE</code> , <code>no</code> , or <code>0</code> , the <code>fillval</code> and/or <code>specval</code> do not apply to HTML output, only to the Edit window display and print/PDF output. NOTE: <code>html-only</code> is useful because the formatting produced by File→Save as HTML... may need some tweaking.	See Figure 64 Editor-only, print-only, html-only on page 104.
ACL or environment variable	When an ACL or environment variable is specified for <code>attname</code> , the variable's value is tested or used instead of an attribute's value.	Figure 63 fillval, SYSTEM-VAR, and environment variable on page 104

Attval keywords

Keywords used in the `attval` characteristic are described below.

#ANY, #NONE, and #NONZERO

[OS B.3.6.5.2.3, Tutorial 1–13]

`#ANY`, `#NONE`, and `#NONZERO` are coded as the value of `attval` in the `specval` category, as detailed in the following table.

Table 8 #ANY, #NONE, #NONZERO

attval keyword	Meaning	Example
<code>#ANY</code>	<code>Specval</code> evaluates to true when <code>attname</code> has a value in the source document.	<code>specval attloc="document" attname="pubnumber" attval=#ANY"</code>
<code>#NONE</code>	<code>Specval</code> evaluates to true when <code>attname</code> does not have a value in the source document, which means the attribute is <code>#IMPLIED</code> in the DTD.	<code>specval attloc="document" attname="pubnumber" attval=#NONE"</code>
<code>#NONZERO</code>	<code>Specval</code> evaluates to true when <code>attname</code> does not contain <code>0</code> , which means the it is type <code>NUMBER</code> in the DTD.	<code>specval attloc="document" attname="changelevel" attval=#NONZERO</code>

**#EQ#, #NE#, #GE#,
#GT#, #LE#, #LT#**

[OS B.3.6.5.2.3, Tutorial 1–15]

The #EQ#, #NE#, #GE#, #GT#, #LE#, #LT# keywords can be used in `attval` to test the value of `attname` with `specval`, as described in the following table. Some examples follow the table.

Table 9 #EQ#, #NE#, #GE#, #GT#, #LE#, #LT#

attval keyword	Meaning
#EQ#	The value of <code>attname</code> must be equal to the value in <code>attval</code> .
#GE#	The value of <code>attname</code> must be greater than or equal to the value in <code>attval</code> .
#GT#	The value of <code>attname</code> must be greater than the value in <code>attval</code> .
#LE#	The value of <code>attname</code> must be less than or equal to the value in <code>attval</code> .
#LT#	The value of <code>attname</code> must be less than the value in <code>attval</code> .
#NE#	The value of <code>attname</code> must be not equal to the value in <code>attval</code> .

When the document attribute is declared in the DTD as CDATA, the text to be tested must be surrounded by literal delimiters (`\`). For example:

```
<specval attname="partnumber" attval="#EQ#\yes\ ">
<specval attname="partnumber" attval="#NE#\no\ ">
<specval attname="partnumber" attval="#GT#\0\ ">
<specval attname="partnumber" attval="#GE#\1\ ">
<specval attname="partnumber" attval="#LT#\100\ ">
<specval attname="partnumber" attval="#GE#\99\ ">
```

When the attribute is declared in the DTD as type NUMBER, literal delimiters are not used. For example:

```
<specval attname="partnumber" attval="#EQ#0">
<specval attname="partnumber" attval="#NE#1">
<specval attname="partnumber" attval="#GT#9999">
<specval attname="partnumber" attval="#GE#10000">
<specval attname="partnumber" attval="#LT#10000">
<specval attname="partnumber" attval="#LE#9999">
```

#ITEM#

[OS B.3.6.5.2.3]

#ITEM# is used to match literal text coded in the FOSI with individual words in an attribute. This is useful for elements that have a *role* or other attribute in

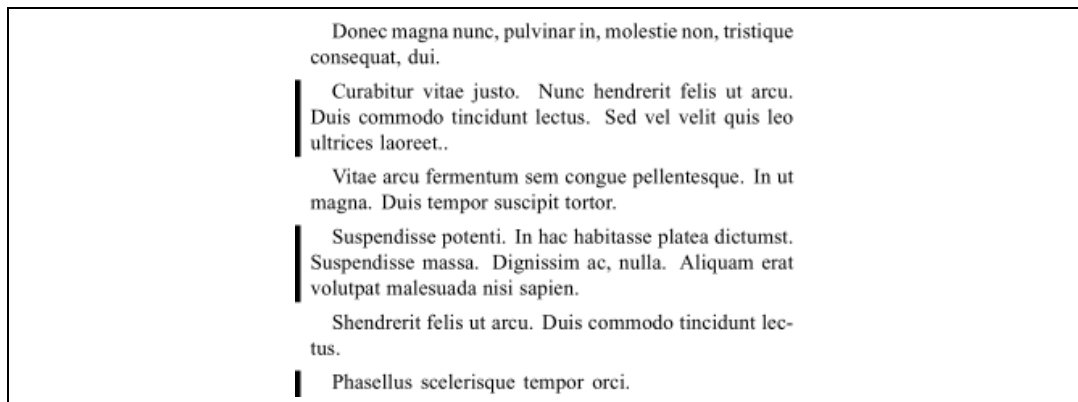
which users enter the desired formatting, such as `center` or `blue underline`. As **Figure 66** demonstrates, different formatting categories can be involved.

Attribute rule keyword examples

This example uses `#EQ#` to compare the value of a CDATA attribute to a string variable. When the two are the same, a change bar is output in the margin.

NOTE: `stringdecl` is necessary for the variable to be recognized as a string.

Figure 56 Compare a CDATA attribute to a string



XML fragment

```
<document level="2">
<para level="1">Donec magna ...</para>
<para level="2">Curabitur vitae ...</para>
<para level="1">Vitae arcu ...</para>
<para level="2">Suspendisse ...</para>
<para level="1">Shendrerit ...</para>
<para level="2">Phasellus ...</para>
...
```

FOSI fragment

```
<stringdecl textid="document-level.txt">
...
<e-i-c gi="document">
<charlist inherit="1"></charlist>
<att>
<fillval attname="level" attloc="document" fillcat="savetext"
fillchar="conrule">
<charsubset>
<savetext textid="document-level.txt">
...
```

```

<e-i-c gi="para">
<charlist inherit="1" charsubsetref="block prespace">
<indent inherit="1" firstln="1em">
</charlist>
<att>
<specval attname="level" attloc="para" attval="#EQ#document-level.txt">
<charsubset>
<chgmark barthick="3pt" baroffset="6pt" type="content"></chgmark>
...

```

An attribute with type NUMBER can be tested against a FOSI counter, as illustrated below. In this example, if the *code* attribute on <section> is less than the *code* attribute on <document>, the <section> is suppressed. If the *code* attribute on <section> is greater than the *code* attribute on <document>, the <section> is suppressed and “ERROR CONDITION” is output.

Figure 57 Compare a NUMBER attribute to a counter

DTD fragment

```

<!ATTLIST document code NUMBER ... >
<!ATTLIST section code NUMBER ... >

```

FOSI fragment

```

<counter initial="0" style="arabic" enumid="document-codect">
...
<e-i-c gi="document">
<charlist inherit="1">...</charlist>
<att>
<fillval attname="code" attloc="document" fillcat="enumerat"
fillchar="inrem">
<charsubset>
<enumerat enumid="document-codect" setvalue="1">
...
<e-i-c gi="section">
<charlist inherit="1"></charlist>
<att>
<specval attname="code" attloc="section" attval="#LT#document-codect">
<charsubset charsubsetref="SUPPRESS"></charsubset>
</att>
<att>
<specval attname="code" attloc="section" attval="#GT#document-codect">
<charsubset charsubsetref="SUPPRESS">
<usetext source="\ERROR CONDITION\">
<subchars charsubsetref="message-format"></subchars>
...

```

In the next example, SYSTEM-FUNC calls an ACL function that performs a substring operation on the contents of the <item> element and returns the first

three letters to the FOSI, which saves it to a string variable. In the page header on verso pages, the FOSI outputs the first three-letter string that appears on the verso page. In the page header on recto pages, the FOSI outputs the last three-letter string that appears on the recto page.

Figure 58 SYSTEM-FUNC substring for page headers

<u>Don</u>	<u>Rae</u>
Donec nun cellentesque libero dolor, ornare non. Etiam vel tortor id massa luctus feugiat.	Hellentesque dapibus dui. Phasellus vitae nisl. In hac habitasse platea dictumst. Cras aliquet, libero vitae pellentesque.
Duis ligula neque, fermentum eu, dictum in, fringilla quam.	Integer pellentesque dapibus dui. Phasellus vitae nisl. In hac habitasse platea dictumst.
Etiam vel tortor id massa luctus feugiat. Duis ligula neque, fermentum eu, dictum in, fringilla quam.	Nun cellentesque libero dolor, ornare non. Etiam vel tortor id massa luctus feugiat. Duis ligula neque, fermentum eu, dictum in, fringilla quam.
Fusce ut nulla a ipsum viverra elementum. Tistique ac, volutpat vel, uma, blandit sed. Phasellus scelerisque tempor orci. Suspendisse ante libero. Pellen Esque libero dolor, ornare non.	Qui velit gravida massa, ut cursus tortor erat vitae massa. Suspendisse pharetra est at neque. To pharetra est at neque hasellus scelerisque.
Gravida massa, ut cursus tortor erat vitae massa. Suspendisse pharetra est at neque. To pharetra est at neque hasellus.	Rasellus scelerisque tempor orci. Suspendisse ante libero.
2	3

XML fragment

```
...
<item>Donec nun cellentesque libero dolor, ornare non...</item>
<item>Duis ligula neque, fermentum eu, dictum in...</item>
<item>Etiam vel tortor id massa luctus feugiat...</item>
<item>Fusce ut nulla a ipsum viverra elementum...</item>
<item>Gravida massa, ut cursus tortor erat vitae massa...</item>
<item>Hellentesque dapibus dui. Phasellus vitae nisl...</item>
<item>Integer pellentesque dapibus dui. Phasellus...</item>
<item>Nun cellentesque libero dolor, ornare non...</item>
<item>Qui velit gravida massa, ut cursus tortor erat...</item>
<item>Rasellus scelerisque tempor orci. Suspendisse...</item>
...
```

ACL function

```
function first_three_letters(window,oid)
{return substr(oid_content(oid),1,3)}
```

FOSI fragment

```
<stringdecl textid="three_letters.txt">
...
```

```

<pageset id="document.page">
<rectopg width="16pi" nomdepth="24pi">
<pageres>...</pageres>
<pagespec pgid="document.recto0">...</pagespec>
</rectopg>
<rectopg width="16pi" nomdepth="24pi">
<pageres>...</pageres>
<pagespec pgid="document.recto1">
...
<header nomdepth="15pt" spaflow="12pt">
<usetext source="three_letters.txt[BO]">
<subchars charsubsetref="block right bold"></subchars>
</usetext>
<ruling thick="0.5pt" lentype="rel" rellen="col" type="single">
<leading inherit="1" lead="3pt">
<textbrk startln="1" endln="1">
</ruling>
</header>
...
<versopg width="16pi" nomdepth="4in">
<pageres>...</pageres>
<pagespec pgid="document.verso1">
...
<header nomdepth="15pt" spaflow="12pt">
<usetext source="three_letters.txt[FI]">
<subchars charsubsetref="block left bold"></subchars>
</usetext>
<ruling thick="0.5pt" lentype="rel" rellen="col" type="single">
<leading inherit="1" lead="3pt">
<textbrk startln="1" endln="1">
</ruling>
</header>
...
<e-i-c gi="item">
<charlist inherit="1" charsubsetref="block prespace">...</charlist>
<att>
<fillval attname="first_three_letters" attloc="SYSTEM-FUNC"
fillcat="savetext" fillchar="conrule">
<charsubset>
<savetext textid="three_letters.txt">
</charsubset>
</att>
...

```

In the following example, #FOSI is used to determine if an optional <title> is present. If so, generated paragraph numbering is output before <title>. If <title> is not present, the generated numbering is output before the paragraph.

Figure 59 Paragraph numbering with optional <title>

```

1. Introduction
Xxx xx xxx xxxx xxx xxx xxxx xxxxx xx xxxx xx xxx xxx xx xxxxx
xxxxxxx. xxx xxxxx xxxxx xxx xxxxxxx xxx.

2. Discussion
Xxx xx xxx xxxx xxx xxx xxxx xxxxx xx xxxx xx xxx xxx xx xxxxx
xxxxxxx. xxx xxxxx xxxxx xxx xxxxxxx xxxxx xxx.

3. Xxx xx xxx xxxx xxx xxx xxxx xxxxx xx xxxx xx xxx xxx xx xxxxx
xxxxxxx. xxx xxxxx xxxxx xxx xxxxxxx xxxxx xxx xxx xxx. Xxx xx
xxx xxx xxx xxx xxxx xxxxx xx xxxx.

4. Conclusion
Xxx xx xxx xxxx xxx xxx xxxx xxxxx xx xxxx xx xxx xxx xx xxxxx
xxxxxxx.

```

DTD fragment

```

<!ELEMENT para0 (title?,para+)>
<!ELEMENT (title|para) (#PCDATA)>

```

XML fragment

```

<para0>
<title>Introduction</title>
<para>Xxx xx xxx xxxx ...</para>
</para0>
<para0>
<title>Discussion</title>
<para>Xxx xx xxx xxxx xxx ...</para>
</para0>
<para0>
<para>Xxx xx xxx xxxx xxx xxx xxxx ...</para>
</para0>
<para0>
<title>Conclusion</title>
<para>Xxx xx xxx xxxx xxx ... </para>
</para0>

```

FOSI fragment

```

<counter initial="0" style="arabic" enumid="para0ct">
<stringdecl textid="sectionct.txt">
<stringdecl textid="title-exists.yn" literal="n">
...
<e-i-c gi="para">
<charlist inherit="1" charsubsetref="block">
<presp minimum="2pt" nominal="2pt" maximum="2pt" priority="med">
</charlist>
<att>
<specval attname="title-exists.yn" attloc="#FOSI" attval="n">
<charsubset>
<presp minimum="8pt" nominal="8pt" maximum="8pt" condit="discard"

```

```

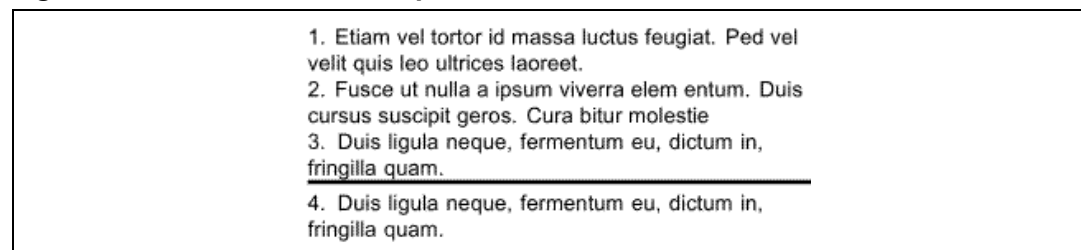
priority="med">
<usetext source="sectionct.txt,\. \ "></usetext>
...
<e-i-c gi="section">
<charlist inherit="1" charsubsetref="block">
<enumerat increm="1" enumid="sectionct">
<savetext textid="sectionct.txt" conrule="sectionct">
<savetext textid="title-exists.yn" conrule="\n\" placemnt="after">
...
<e-i-c gi="title" context="section">
<charlist inherit="1" charsubsetref="block">
<presp minimum="8pt" nominal="8pt" maximum="8pt" condit="discard"
priority="med">
<usetext source="sectionct.txt,\. \ "></usetext>
<savetext textid="title-exists.yn" conrule="\y\">
...

```

In the next figure, #FOSI is used to output a horizontal rule after the third <para>. #FOSI is coded in a pseudo-element that is processed after paract is incremented.

NOTE: If #FOSI is coded directly in the e-i-c for <para> rather than in pseudo-element, the rule is output after the fourth <para>.

Figure 60 Third element outputs horizontal rule



XML fragment

```

<para>Etiam vel tortor ...</para>
<para>Fusce ut nulla ...</para>
<para> Duis ligula neque ...</para>
<para> Duis ligula neque ...</para>

```

FOSI fragment

```

<counter initial="0" style="arabic" enumid="paract">
...
<e-i-c gi="para">
<charlist inherit="1" charsubsetref="block">
<enumerat increm="1" enumid="paract">
<usetext source="paract,\. \ " placemnt="before"></usetext>

```

```

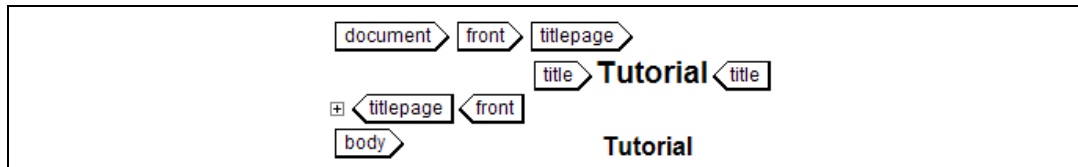
<usetext source="<maybe-ruling.psu>,</maybe-ruling.psu>" placemnt="after">
...
<e-i-c gi="maybe-ruling.psu">
<charlist inherit="1"></charlist>
<att>
<specval attname="paract" attloc="#FOSI" attval="3">
<charsubset>
<ruling thick="2pt" lentype="rel" relen="col" placemnt="after"
type="single">
<leading inherit="1" lead="3pt">
<textbrk startln="1" endln="1">
...

```

#XPATH is used in this example to output the contents of an element from a different element. The graphic shows the Edit window display.

NOTE: #XPATH and an attribute rule are not required for this. The same thing can be output by saving the <title> element contents to a string variable for output from <body>.

Figure 61 #XPATH with fillval



FOSI fragment

```

<e-i-c gi="body">
<charlist inherit="1" charsubsetref="block">...</charlist>
<att>
<fillval attname="/document/front/titlepage/title" attloc="#XPATH"
fillcat="usetext" fillchar="source">
...

```

The next example also uses #XPATH. When the *code* attribute on the <section> tags matches the code attribute on the <document> tag, the <section> is suppressed. attloc="#XPATH" causes attname to be evaluated as an XPath expression.

NOTE: Native FOSI can be used for this purpose instead of XPath. See **Figure 56 Compare a CDATA attribute to a string** on page 97 and **Figure 57 Compare a NUMBER attribute to a counter** on page 98 for examples.

FOSI Tip

Attribute rules with and without #XPATH are evaluated at runtime, whereas the rest of FOSI code is compiled. Consequently, a FOSI with many attribute rules and runtime operations takes longer to format documents. Too many can slow the formatter to a crawl.

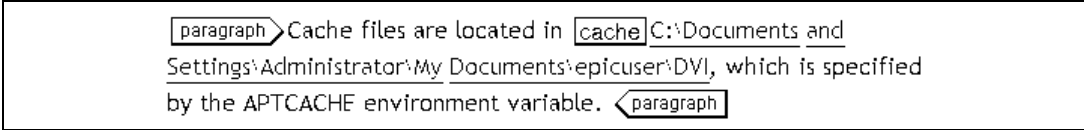
Figure 62 Attloc="#XPATH"

```

<e-i-c gi="section">
<charlist inherit="1"></charlist>
<att>
<specval attname="@code=ancestor::document/@code"
attloc="#XPATH" attval="1">
<charsubset charsubsetref="SUPPRESS">
...

```

In this example, the value of the APTCACHE environment variable is displayed in the Edit window. Its value is a path to cache files, which the FOSI uses as a link. Double-clicking on the gentext in the Edit window opens a browser window for the designated directory.

Figure 63 fillval, SYSTEM-VAR, and environment variable


Cache files are located in C:\Documents and Settings\Administrator\My Documents\epicuser\DVI, which is specified by the APTCACHE environment variable.

FOSI fragment

```

<e-i-c gi="cache">
<charlist inherit="1" charsubsetref="inline"></charlist>
<att>
<fillval attname="APTCACHE" attloc="SYSTEM-VAR" fillcat="link"
fillchar="href">
<fillval attname="APTCACHE" attloc="SYSTEM-VAR" fillcat="usetext"
fillchar="source">
<charsubset>
<link>
<usetext>
<subchars charsubsetref="underline"></subchars>
...

```

The next example shows the use of editor-only, print-only, and html-only. In the Edit window, the element is not indented. In print/PDF, the element is indented 2pi on the left. In HTML output, the element is indented 3pi on the left and right.

Figure 64 Editor-only, print-only, html-only

```

<att>
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="yes">
<charsubset>
<indent inherit="1" firstln="*" leftind="0pt" rightind="0pt">
...
<att>

```

```

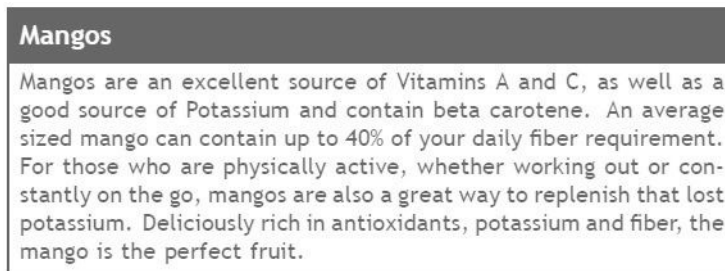
<specval attname="print-only" attloc="SYSTEM-VAR" attval="yes">
<charsubset>
<indent inherit="1" firstln="*" leftind="2pi" rightind="0pt">
...
<att>
<specval attname="html-only" attloc="SYSTEM-VAR" attval="yes">
<charsubset>
<indent inherit="1" firstln="*" leftind="3pi" rightind="3pi">
...

```

The following example illustrates using an attribute value from a previous sibling element. A sibling's attribute value cannot be directly accessed with an attribute rule the way an ancestor element can be. In this case, the color for the document is specified in the <title> *color* attribute. It is used as the font color for <description> as well as the background color for <title>.

NOTE: The color specified in this example is a yellow-orange, but the color appears as gray here.

Figure 65 Fillval with #FOSI



DTD fragment

```

<!ELEMENT group (title,description)>
<!ELEMENT (title|description) (#PCDATA)>
<!ATTLIST title color CDATA #REQUIRED>

```

XML fragment

```

<group>
<title color="#FFCC00">Mangos</title>
<description>Mangos are ... the perfect fruit.
</description>
</group>

```

FOSI fragment

```

<stringdecl textid="color.txt">
...
<e-i-c gi="description" context="group">

```

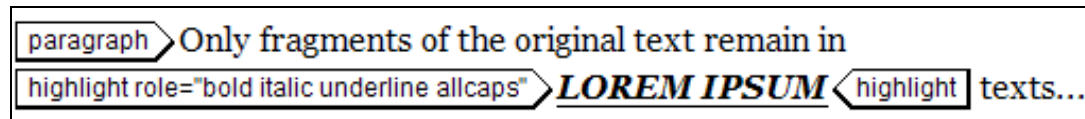
```

<charlist inherit="1" charsubsetref="block"></charlist>
<att>
<fillval attname="color.txt" attloc="#FOSI" fillcat="boxing"
fillchar="outclr">
<fillval attname="color.txt" attloc="#FOSI" fillcat="highlt"
fillchar="fontclr">
<charsubset>
<highlt inherit="1">
<boxing toffset="6" boffset="6pt" loffset="6pt" roffset="6pt"
trel="top" brel="bottom" siderel="col" thick="1pt "
ttype="tsingle" btype="bsingle" ltype="lsingle" rtype="rsingle">
...
<e-i-c gi="title" context="group">
<charlist inherit="1" charsubsetref="title">
...
<savetext textid="color.txt" conrule="#CONTENT(color)">
...
<att>
<fillval attname="color" attloc="title" fillcat="boxing" fillchar="outclr">
<fillval attname="color" attloc="title" fillcat="boxing" fillchar="inclr">
<charsubset charsubsetref="bold">
<boxing toffset="8pt" boffset="6pt" loffset="6pt" roffset="6pt"
trel="top" brel="bottom" siderel="col" thick="1pt"
ttype="tsingle" btype="bsingle" ltype="lsingle" rtype="rsingle">
...
<att>
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#ANY">
<fillval attname="color" attloc="title" fillcat="highlt" fillchar="bckclr">
<charsubset>
<highlt inherit="1" fontclr="#FFFFFF">
...
<att>
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#NONE">
<charsubset charsubsetref="allcaps">
<highlt inherit="1" fontclr="#FFFFFF">
...

```

In this example of #ITEM#, the <highlight> element is shown in the Edit window. The values for the *role* attribute are used in the font and highlt categories.

Figure 66 specval with #ITEM#



FOSI fragment

```

<e-i-c gi="highlight">
<charlist inherit="1"></charlist>

```

```

<att>
<specval attname="role" attloc="highlight" attval="#ITEM#bold">
<charsubset>
<font inherit="1" weight="bold">
...
<att>
<specval attname="role" attloc="highlight" attval="#ITEM#italic">
<charsubset>
<font inherit="1" posture="italic">
...
<att logic="or">
<specval attname="role" attloc="highlight" attval="#ITEM#underline">
<specval attname="role" attloc="highlight" attval="#ITEM#underscore">
<charsubset>
<highlt inherit="1" scoring="1" scorespc="1">
...
<att>
<specval attname="role" attloc="highlight" attval="#ITEM#allcaps">
<charsubset>
<highlt inherit="1" allcap="1">
...

```

Sorted, merged, grouped material (indexing) keywords

Two keywords can be used with the `locmarkuppriority useparam`: `#NONE` and `#OTHER`

- `#NONE` applies to locators where no markup is specified.
- `#OTHER` applies to all other markup found around a locator, but not specified in the list of element names.

NOTE: If `#OTHER` is not included in the value of the `locmarkuppriority useparam`, element names found around a page locator, but not mentioned explicitly in the list of element names, generate an error message.

Table 15 Locmarkuppriority examples on page 164 provides examples of the use of these keywords.

Significant record ends

Following is what the FOSI standard [O.S. B.3.14] has to say on the subject of significant record ends.

Although the SGML standard (ISO 8879) carefully defines which record ends (“carriage returns” or “line breaks”) in the input source file are significant and which are to be ignored by the parser, it does not prescribe what the application should do with significant record ends.

Formatting applications in compliance with MIL-PRF-28001 should normally treat a significant record end as a space character; furthermore, consecutive multi-space characters (including record ends being treated as space characters) should normally be treated for the purpose of composition as a single space character. The exception is in the case of an e-i-c with “asis” quadding. In this case, each space affects formatting and each significant record end causes a line break during formatting (consecutive record ends should produce multiple line breaks, that is, they should produce a “blank line” in the composed output). In “asis” mode, it is an input error if the content given between two record ends will not fit on one output line. What happens in this case is left to the output system.

In Arbortext Editor, a line overset condition results when an “asis” line does not fit on one output line. Whether than condition is reported depends on formatting fault settings (see **Formatting faults** on page 727 for details).