Any sufficiently advanced technology is indistinguishable from magic.
—Arthur C. Clarke

CHAPTER 8
How To Get There from Here

FOSI has been in use for more than twenty years. Many formatting challenges have been solved along the way. This chapter contains specific how-to information so you don't have to re-invent the wheel.

How to output PDF links, bookmarks, document properties, and viewer settings

Arbortext Editor supports generation of PDF links, bookmarks, document properties, and viewer settings from SGML and XML documents. They are all discussed below.

▷ PDF links, page 561
▷ PDF bookmarks, page 567
▷ PDF document properties, page 575
▷ PDF viewer settings, page 576

NOTE: These techniques apply only to the direct PDF method.

PDF links

Automatically generated, FOSI-generated, and manually authored PDF links are discussed below.

Automatically generated PDF links

Publishing directly to PDF automatically creates links that go from:

- Each entry in the Table of Contents to the page on which that entry appears
- A cross reference to the page on which the cross-referenced element appears
- Each index entry to its corresponding page number

NOTE: Automatically generated PDF links can be disabled, as shown on page 567.

FOSI-generated PDF links

The `link` category (page 477) is used to generate PDF links to:

- A named destination
- A URL

`Link` can be used in different ways:

- In the resolved `charlist` of an `e-i-c`.

  The `href` characteristic specifies a hard-coded URL or a hard-coded ID that presumably is included in the document.
- In a `fillval charsubset`.

  An attribute on an element in the document specifies an ID or URL, which the `fillval` category assigns to the `href` characteristic on the `link` category.
- In `usetext subchars`.

  Clicking on the gentext output by the `usetext source` activates the `link` category coded in the `usetext subchars`.

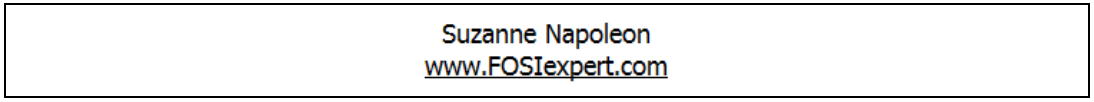
FOSI-generated PDF links examples

See **Figure 274 Hard-coded link in Edit window** on page 477 for an example of `link` coded in a `charlist`.

See **Figure 275 Attribute-based link in Edit window** on page 478 for an example of `link` coded in an attribute rule.

In the following example, the `link` is hard-coded in the `subchars` for the generated text. The image shows the PDF output.

Figure 316 Gentext is hard-coded link



XML fragment

```
<author>Suzanne Napoleon</author>
```

FOSI fragment

```
<e-i-c gi="author">
<charlist inherit="1" charsubsetref="block center">
<usetext source="\www.FOSIexpert.com\" placemnt="after">
<subchars charsubsetref="block blue underline">
<link href="http://www.fosiexpert.com/index.html">
...
```

Authored PDF links

Three ways of authoring PDF links are discussed in this section.

DCF-declared PDF link and target elements

An element declared as a link in the DCF (doctype configuration file) can be inserted in a document using Insert▸Link.... An element declared as a target in the DCF can be inserted with Insert▸Link Target.... A declared link element must have an attribute to specify its target. A declared target element must have an ID attribute to be used by the linking element.

A DCF-declared link can specify:

- Linking within a PDF file to a named destination.
- Linking to a named destination or a specific page in another PDF file.
- Linking to a URL or URL anchor.
- Linking to an element declared in the DCF as a target.

Please see Arbortext Editor documentation for information on declaring and inserting these links.

Links within a PDF file

A link within a PDF file (internal link) is specified by a named destination created by an ID on an element in the source document. These links are also functional in the Edit window.

Clicking on an internal link in a PDF file loads the relevant page in the PDF viewer. Clicking on an internal link in the Edit window moves the cursor to the element with the matching ID.

Links from one PDF file to another

Clicking on a link to another PDF file (external link) loads the target PDF file in the PDF viewer. The opening position of the target file can be specified in two ways:

- As a page number (`#page=`), which displays the specified page in the PDF viewer window.
- As a named destination (`#nameddest=`) created by an ID in the document, which positions the named destination at the top of the PDF viewer window.

PDF links to a URL

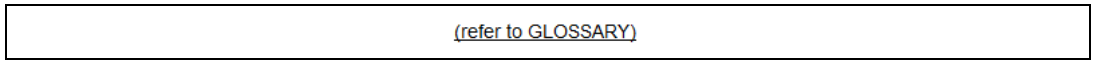
Clicking on a URL link in a PDF viewer opens a browser window that displays the specified page, assuming it is available. If a URL anchor is specified, it is displayed at the top of the browser window.

Authored PDF links examples

These examples use the same DTD and DCF fragments.

In the first figure, the link is to a named destination in the same PDF file. This link is also functional in the Edit window.

Figure 317 Authored internal PDF link



XML DTD fragment

```
<!ELEMENT pdflink ANY>
<!ATTLIST pdflink linkto ID #IMPLIED>
```

DCF fragment

```
<Specials>
<Link element="pdflink" idref="linkto" primary="yes"/>
```

XML fragment

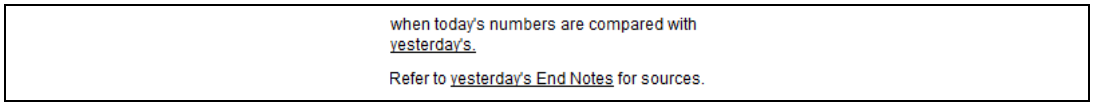
```
<para>...(refer to <pdflink linkto="glossary">Glossary</pdflink>).
...
<appendix id="glossary">
<title>Glossary</title>
...
```

FOSI fragment

```
<e-i-c gi="pdflink">
<charlist inherit="1" charsubsetref="blue underline allcaps"></charlist>
...
```

In the next figure, the link is to a named destination in another PDF file.

Figure 318 Authored external PDF links



XML fragment — Current document

```
<para>...when today's numbers are compared with <pdflink linkto=
"yesterday.#page=2">yesterday's</pdflink>.</para>
<para>Refer to <pdflink linkto="yesterday.pdf#nameddest=endnotes">
yesterday's End Notes</pdflink> for sources.</para>
```

XML fragment — External document

```
<appendix id="endnotes">
<title>End Notes</title>
...
```

FOSI fragment

```
<e-i-c gi="pdflink">
<charlist inherit="1" charsubsetref="blue underline"></charlist>
...
```

The last example shows a link from a PDF file to a URL.

Figure 319 Authored link to a URL

XML fragment

```
<item>FOSI resource: <pdflink linkto="http://www.FOSIexpert.com">
FOSIexpert.com</pdflink></item>
```

FOSI fragment

```
<e-i-c gi="pdflink">
```

```
<charlist inherit="1" charsubsetref="gray underline"></charlist>
</e-i-c>
```

<_link> and <_target> PIs for PDF links

<_link> and <_target> hyperlinks can be inserted in a document by selecting Tools▸Create Hyperlink..., as described in **Link and target processing instructions** on page 714.

NOTE: <_link> and <_target> are being deprecated in Arbortext Editor.

<_link> and <_target> examples

See **Link and target examples** on page 716 for examples.

PDF link-related ACL

The following environmental variables affect PDF linking:

- APTNOPDFLINKS controls whether links are created in PDF files. The default is `no`. `Yes` disables links, although bookmarks for a Table of Contents and a List of Tables are still created.

  NOTE: The APTNOPDFLINKS environment variable has no effect on HTML output.
- APTNAMEDDESTTOPAGE set to any value converts named destinations in PDF links to page numbers.

PDF bookmarks

Bookmarks can be automatically generated in two ways that are not mutually exclusive.

- Based on declarations in the DCF file
- Based on <atidmd> namespace markup (page 773) coded in the FOSI

These are covered in the following sections.

▷ DCF-based PDF bookmarks, page 568
▷ FOSI-based PDF bookmarks, page 573

NOTE: You can disable automatically generated PDF bookmarks by coding the FOSI to generate the following at the beginning of the document:

```
<atidmd:DocumentMetaData source="atend">
<atidmd:DocView mode="bookmarks" fit="fitPage" destination="">
</atidmd:DocView>
<para></para>
</atidmd:DocumentMetaData>
```

DCF-based PDF bookmarks

When a PDF file is created, bookmarks are automatically generated according to `ElementOption` settings in the doctype's DCF (doctype configuration file). Specifically, every element that is declared as the title of an element that is declared as division is bookmarked in the PDF file.

Nesting of PDF bookmarks follows the hierarchy of the document structure. Table titles, however, are listed after the hierarchical bookmarks.

The first block of text in the title is bookmarked for PDF, which could be generated text rather than authored text. To bookmark authored text instead, code the gentext on the division `e-i-c` instead of on the title `e-i-c`, as shown in **Figure 322**.

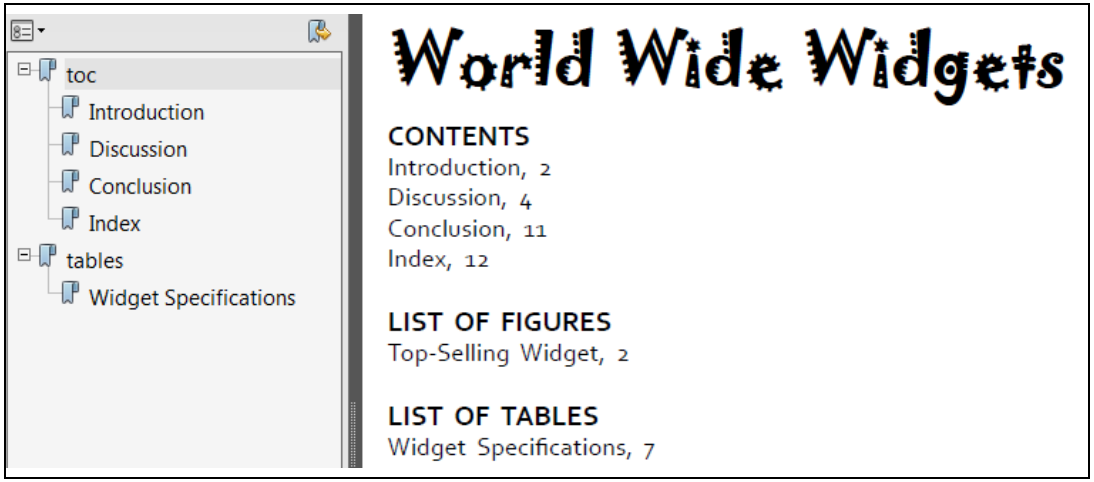
If a title is suppressed and there is no title gentext, the first available text is output as the bookmark.

DCF-based bookmarks examples

In the first example of DCF-based bookmarks, <index> is a singleton that is declared in the DCF as a division and <title> is declared as a title. The `e-i-c` for <index> outputs a <title> (as a pseudo-element), which creates a PDF bookmark for it.

The figures show the PDF bookmark and the document Table of Contents side by side for purposes of comparison. Notice the automatically created Tables bookmark. However, there is no bookmark for the List of Figures shown in the TOC. To create a bookmark for each <figure> as it occurs in the document structure, declare the element in the DCF file as a division and its title or caption as a title.

Figure 320 <Title> elements and pseudo-element generate PDF bookmarks



XML DTD fragment

```
<!ELEMENT document (title,front,body,index)
<!ELEMENT body (chapter+)>
<!ELEMENT chapter (title,...)>
<!ELEMENT front (...,toc,...)>
<!ELEMENT index EMPTY>
<!ELEMENT toc EMPTY>...
```

DCF fragment

```
<ElementOptions>
<ElementOption category="division" element="document"/>
<ElementOption category="division" element="index"/>
<ElementOption category="division" element="chapter"/>
<ElementOption category="title" element="title"/>
</ElementOptions>
```

XML fragment

```
<document><title>World Wide Widgets</title>
<front>...<toc/>...</front>
<body>
<chapter><title>Introduction</title>
...
<figure>
<graphic .../>
<caption>Top-Selling Widget</caption>
</figure>
...
</chapter>
<chapter><title>Discussion</title>
...
```

```
<table><title>Widget Specifications</title>...</table>
...
</chapter>
<chapter><title>Conclusion</title>...</chapter>
</body>
<index/>
</document>
```

FOSI fragment

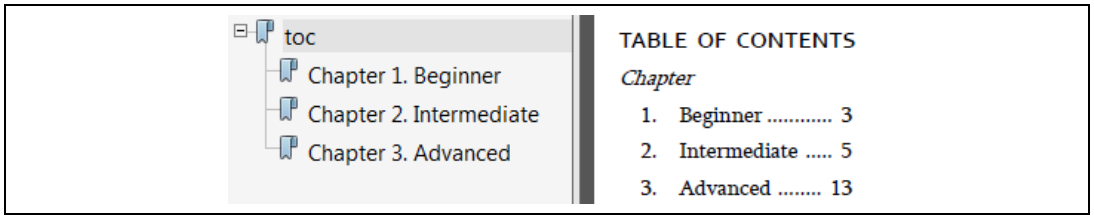
```
<stringdecl textid="caption.txt" hotlink="1">
<stringdecl textid="document-title.txt" hotlink="1">
<stringdecl textid="index.app">
<stringdecl textid="lof.app" status="1" hotlink="1">
<stringdecl textid="lot.app" status="1" hotlink="1">
<stringdecl textid="chapter-title.txt" hotlink="1">
<stringdecl textid="table-title.txt" hotlink="1">
<stringdecl textid="toc.app" status="1" hotlink="1">
...
<e-i-c gi="caption" context="figure">
<charlist inherit="1" charsubsetref="caption">
<savetext textid="caption.txt" conrule="#CONTENT">
<savetext textid="lof.app" append="1"
conrule="<toc.fmt>,caption.txt,\, \,folioct.txt[BO],</toc.fmt>">
...
<e-i-c gi="index">
<charlist inherit="1">
<usetext source="<title>,\Index\,</title>">
<subchars charsubsetref="title next-page">
...
<usetext source="index.app" userule="2"></usetext>
...
<e-i-c gi="title" context="document">
<charlist inherit="1" charsubsetref="document-title">
<usetext source="\Contents\" placemnt="after">
<subchars charsubsetref="title allcaps">
...
<usetext source="toc.app" placemnt="after"></usetext>
<usetext source="\List of Figures\" placemnt="after">
<subchars charsubsetref="title allcaps">
...
<usetext source="lof.app" placemnt="after"></usetext>
<usetext source="\List of Tables\" placemnt="after">
<subchars charsubsetref="title allcaps">
...
<usetext source="lot.app" placemnt="after">
...
<e-i-c gi="title" context="index">
<charlist inherit="1" charsubsetref="title">
<savetext textid="toc.app" append="1"
conrule="<toc.fmt>,\Index, \,folioct.txt[BO],</toc.fmt>">
...
```

```
<e-i-c gi="title" context="chapter">
<charlist inherit="1" charsubsetref="title">
<savetext textid="chapter-title.txt" conrule="#CONTENT">
<savetext textid="toc.app" append="1"
conrule="<toc.fmt>,chapter-title.txt,\, \,folioct.txt[BO],</toc.fmt>">
<savetext textid="index.app" append="1"
conrule="<ixpt>,chapter-title.txt,</ixpt>,<ixpgstr>,folioct.txt[BO],
</ixpgstr>,<ixpgkey>,folioct.txt[BO],</ixpgkey>">
</charlist>
<att>
<fillval attname="id" attloc="chapter"
fillcat="savetext" fillchar="textid">
<charsubset>
<savetext conrule="chapter-title.txt" xrefidtag="chapter">
...
<e-i-c gi="title" context="table">
<charlist inherit="1" charsubsetref="title">
<savetext textid="table-title.txt" conrule="#CONTENT">
<savetext textid="lot.app" append="1"
conrule="<toc.fmt>,table-title.txt,\, \,folioct.txt[BO],</toc.fmt>">
...
```

The next three figures use the same DTD, DCF, and XML fragments. Also, the `counter` and `stringdecl` categories in the FOSI fragment in **Figure 321** apply to the other FOSI fragments.

The first image shows gentext and <title> content as bookmarks.

Figure 321 PDF bookmark with gentext and content



XML DTD fragment

```
<!ELEMENT chapter (title,...)>
```

DCF fragment

```
<ElementOptions>
<ElementOption category="division" element="chapter"/>
<ElementOption category="title" element="title"/>
</ElementOptions>
```

XML fragment

```
<chapter><title>Beginner</title>...</chapter>
```

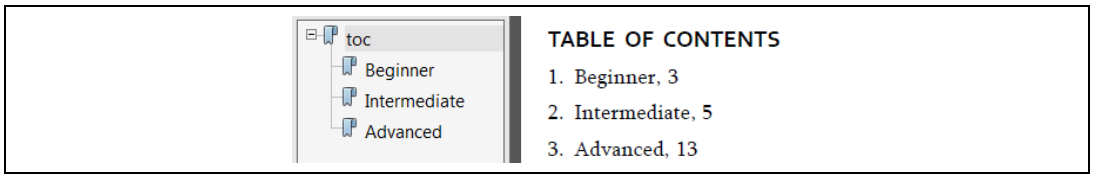
```
<chapter><title>Intermediate</title>...</chapter>
<chapter><title>Advanced</title>...</chapter>
```

FOSI fragment

```
<counter initial="0" style="arabic" enumid="chapterct">
<stringdecl textid="chapterct.txt" hotlink="1">
<stringdecl textid="chapter-title.txt" hotlink="1">
<stringdecl textid="toc.app" status="1" hotlink="1">
...
<e-i-c gi="chapter">
<charlist inherit="1">
<textbrk startpg="recto">
<enumerat increm="1" enumid="chapterct">
<savetext textid="chapterct.txt" conrule="chapterct">
...
<e-i-c gi="title" context="chapter">
<charlist inherit="1" charsubsetref="title endline">
<usetext source="\Chapter \,chapterct.txt,\. \"></usetext>
<savetext textid="chapter-title.txt"
conrule="chapterct.txt,\. \,#CONTENT">
<savetext textid="toc.app" append="1"
conrule="<toc.fmt>,chapter-title.txt,\, \,folioct.txt[BO],</toc.fmt>">
...
```

In the following example, the gentext is output from the division element (<chapter>), so it does not appear in the bookmarks.

Figure 322 PDF bookmark without gentext



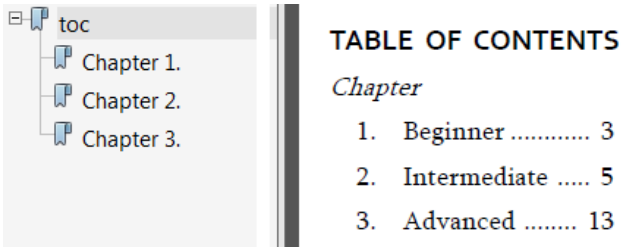

```
<e-i-c gi="chapter">
<charlist inherit="1">
<enumerat increm="1" enumid="sectionct">
<savetext textid="chapterct.txt" conrule="chapterct">
<usetext="chapterct.txt,\. \"></usetext>
...
<e-i-c gi="title" context="chapter">
<charlist inherit="1" charsubsetref="title endline">
<savetext textid="chapter-title.txt" conrule="chapterct.txt,\. \,#CONTENT">
<savetext textid="toc.app" append="1"
conrule="<toc.fmt>,chapter-title.txt,\, \,folioct.txt[BO],</toc.fmt>">
...
```

In the last example, the <title> is suppressed and doesn't appear in the bookmarks. However, the gentext is used.

TIP

`Set gentexttagdisplay=full` displays <atidmd> tags in the Edit window.

Figure 323 PDF bookmark with only gentext




```
<e-i-c gi="title" context="chapter">
<charlist inherit="1" charsubsetref="title endline SUPPRESS">
<savetext textid="chapter-title.txt" conrule="#CONTENT">
<savetext textid="toc.app" append="1"
conrule="<toc.fmt>,1em,chapterct.txt,\.\,1em,chapter-title.txt,
dotfill,folioct.txt[BO],</toc.fmt>">
<usetext source="\Chapter \,chapterct.txt,\. \"></usetext>
...
<e-i-c gi="chapter">
<charlist inherit="1" charsubsetref="block">
<enumerat increm="1" enumid="chapterct">
<savetext textid="chapterct.txt" conrule="chapterct">
...
```

FOSI-based PDF bookmarks

A FOSI can be coded to output <atidmd> namespace markup configured to generate PDF bookmarks. <atidmd> namespace markup is detailed in **PDF document properties** on page 575.

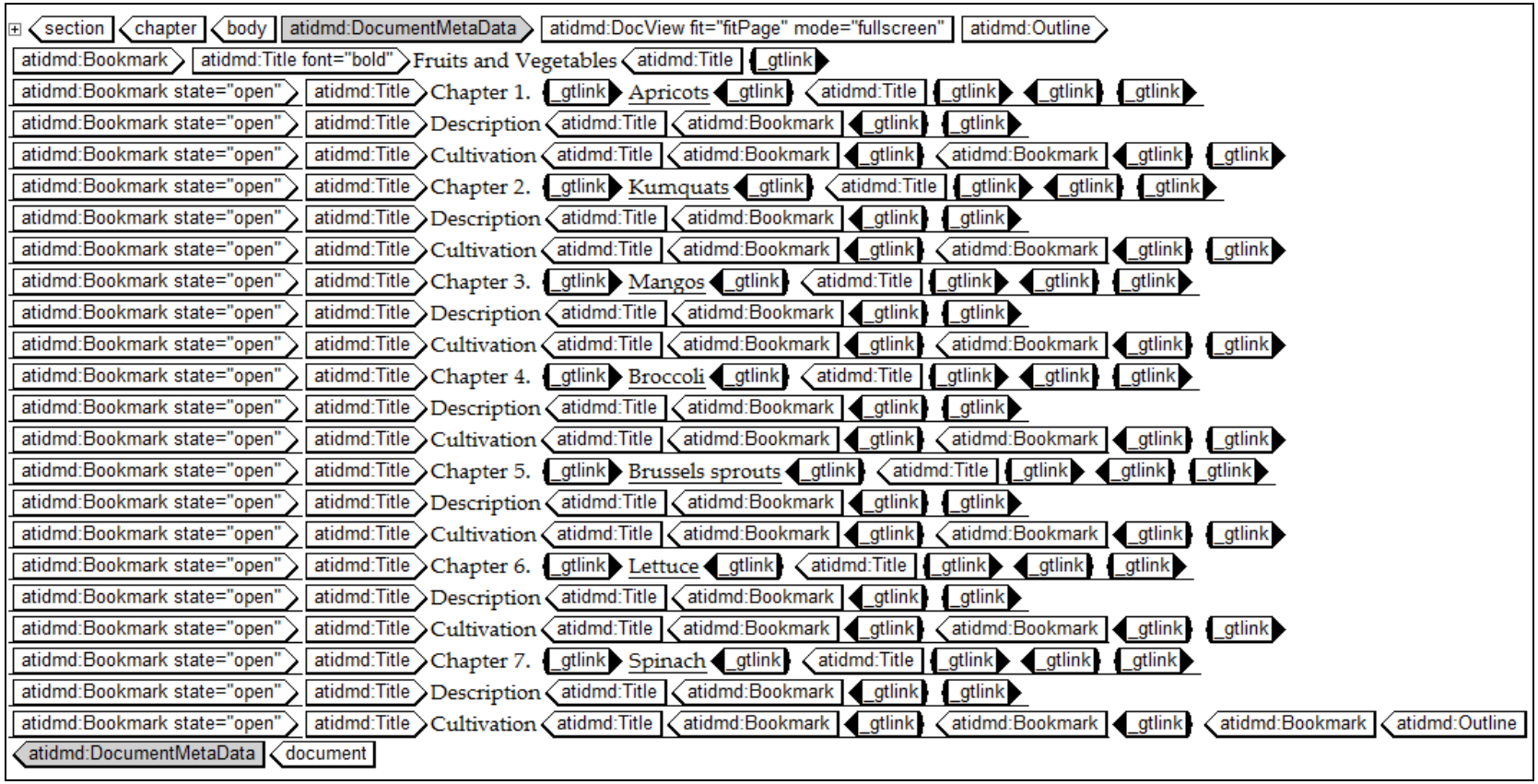
NOTE: When gentext tags are displayed in the Edit window, as shown in **Figure 324**, the <_gtlink> processing instructions that support gentext hyperlinks and PDF links are visible. More information about <_gtlink> is available at **<_gtlink> PI** on page 721.

FOSI-based bookmark examples

Closed and open bookmarks are shown in **PDF customization example** below.

Figure 324 illustrates how <atidmd> markup can be formatted for Edit window display. Notice the <_gtlink> PIs, which support gentext and PDF linking.

Figure 324 Formatted <atidmd> markup in Edit window



FOSI fragment

```
<e-i-c gi="atidmd:Bookmark">
<charlist inherit="1" charsubsetref="startline"></charlist>
</e-i-c>
<e-i-c gi="atidmd:DocumentMetaData">
<charlist inherit="1" charsubsetref="gray">
...
```

PDF document properties

The DocumentMetaData namespace tag <atidmd:DocInfo> is used to specify PDF document properties. It can be coded in a `usetext` for output anywhere in a document, either with other child elements of <atidmd:DocumentMetaData> or separately, as shown in **Figure 325**.

PDF document properties example

In this example, <atidmd> markup specifies the document name, author, and date properties of the PDF file that is created. Notice `placemnt="after"` on the <figure> `e-i-c`, which is needed so the string variables have content.

Figure 325 <atidmd:DocInfo> specifies PDF document properties

XML DTD fragment

```
<!ELEMENT document (front,body)>
<!ElEMENT front (title,author,date)>
```

FOSI fragment

```
<stringdecl textid="author.txt" hotlink="1">
<stringdecl textid="date.txt" hotlink="1">
<stringdecl textid="document-title.txt" hotlink="1">
...
<e-i-c gi="author" context="front">
<charlist inherit="1">
<savetext textid="author.txt" conrule="#CONTENT">
...
<e-i-c gi="date" context="front">
<charlist inherit="1">
<savetext textid="date.txt" conrule="#CONTENT">
...
<e-i-c gi="front">
<charlist inherit="1">
<usetext placemnt="after" source='!<atidmd:DocumentMetaData
bookmarks="stylesheet" source="atend"><atidmd:DocInfo><atidmd:Entry>
<atidmd:Key>Title</atidmd:Key><atidmd:Value>!,document-title.txt,
!</atidmd:Value></atidmd:Entry><atidmd:Entry><atidmd:Key>Author
</atidmd:Key><atidmd:Value>!,author.txt,!</atidmd:Value></atidmd:Entry>
<atidmd:Entry><atidmd:Key>Date</atidmd:Key><atidmd:Value>!,date.txt,
!</atidmd:Value></atidmd:Entry></atidmd:DocInfo></atidmd:DocumentMetaData>
...
<e-i-c gi="title" context="document">
<charlist inherit="1" charsubsetref="title center">
<savetext textid="document-title.txt" conrule="#CONTENT">
...
```

PDF viewer settings

The DocumentMetaData namespace tag <atidmd:DocView> is used to specify PDF window customizations. It can be coded in a `usetext` for output anywhere in a document, separately, as shown in **Figure 325**, or with other child elements of <atidmd:DocumentMetaData>.

PDF window customization example

In this example, <atidmd:DocView> specifies that the document should be scaled to fit within the width of the PDF viewer window and that the content with the named destination of "openpage" should appear at the top of the window.

Figure 326 <Atidmd:DocView> specifies PDF window properties

XML fragment

```
<chapter id="openingpage"><title>Latest Update</title>...
```

FOSI fragment

```
<usetext source='!<atidmd:DocumentMetaData bookmarks="stylesheet"
source="atend"><atidmd:DocView fit="fitWidth" destination="openpage"/>
</atidmd:DocumentMetaData>!'>
```

How to print with crop marks

Crop marks do not have to be coded in the FOSI. Under Options in the print panel, you'll see a checkbox for selecting Crop Marks. Keep in mind that crop marks are located outside the dimensions of the page. So, for crop marks to appear, the size of the output paper must be larger than the page dimensions specified in the FOSI. For example, if you print a 7x9 page to 8½x11 paper and select Crop Marks, the crop marks are printed. But if the document is 8½x11 and the output paper is the same size, crop marks do not appear.

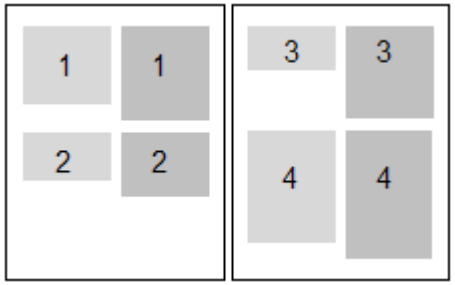
Crop marks and registration marks can be coded in a FOSI, if desired. Appropriate page dimensions and margins must be specified in the `pagesets` in order to obtain the desired page size. See **Figure 201 Page border with `regions`** on page 336 for information on using `regions` to output horizontal and vertical rules.

How to output two languages side by side

Supporting two languages side by side generally uses `algroup` and `indent` categories. It may be tempting to use generated tables, but that approach has drawbacks, as described in **Table formatting limitations** on page 799.

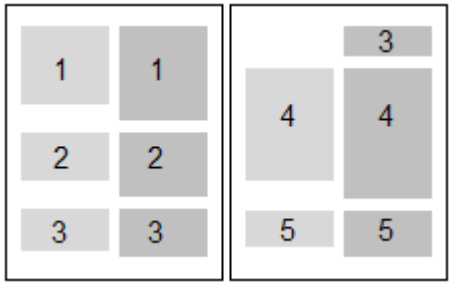
When `algroup` is coded in an `e-i-c`, the content of the element is kept together and does not permit a page break. **Figure 327** below illustrates the short pages that result when side-by-side blocks are unbreakable. This increases the number of pages in the document.

Figure 327 Unbreakable side-by-side blocks



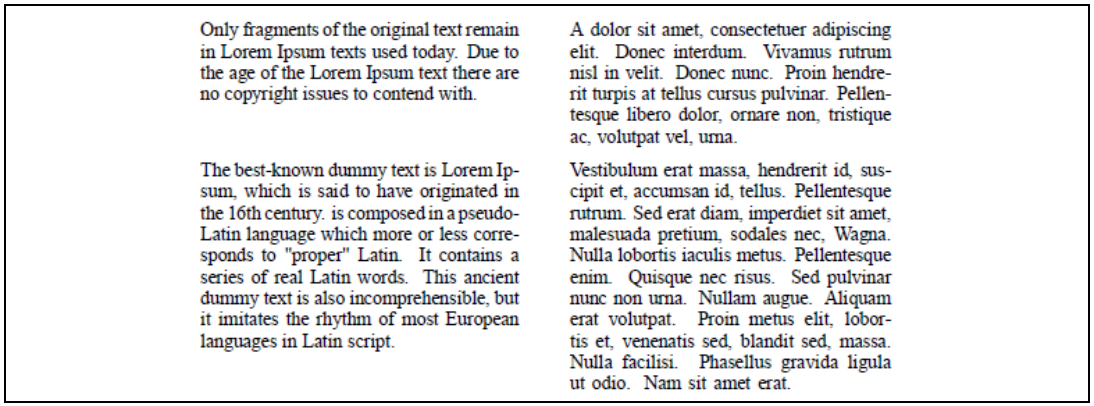
However, only the first `e-i-c` to be aligned requires the `algroup` category. When `algroup` is not coded in the second `e-i-c` to be aligned, page breaks are permitted in the content. To avoid short pages, output the longer language in the second column. Do not code `algroup` in the `e-i-c` for the second column so that page breaks are allowed. **Figure 328** shows how more content fits on a page when the second side-by-side block is allowed to break.

Figure 328 Breakable side-by-side blocks



The next figure shows a paragraph in English and a paragraph of nonsense text aligned side by side. English is specified as the hyphenation language for the first column. French hyphenation is specified for the second column.

Figure 329 side-by-side paragraphs in two languages



XML DTD fragment

```
<!ELEMENT chapter (col1,col2)+>
<!ELEMENT (col1|col2) (#PCDATA)>
```

XML fragment

```
<chapter>
<col1>Only fragments...</col1>
<col2>A dolor sit...</col2>
<col1>The best-known...</col1>
<col2>Vestibulum erat...</col2>
...
```

FOSI fragment

```
<hyphrule language="en" wordbrk="enhyph.exc"
brkafchr="_:\/+-" clbrkok="0" brkalways="1">
<hyphrule language="fr" wordbrk="frhyph.exc"
brkafchr="_:\/+-" clbrkok="0" brkalways="1">...
<e-i-c gi="col1">
<charlist inherit="1" charsubsetref="block prespace">
<hyphen lang="en" hyph="1">
<indent inherit="1" leftind="0" rightind="*+7pi" firstln="*">
<algroup refpoint="first">
...
<e-i-c gi="col2">
<charlist inherit="1" charsubsetref="block prespace">
<hyphen lang="fr" hyph="1">
<indent inherit="1" leftind="9pi" rightind="*+7pi" firstln="*">
...
```

When groups of elements must be aligned side by side, some of the DTD markup and FOSI coding are a little different, as illustrated in the following figure.

Figure 330 side-by-side element groups

XML DTD fragment

```
<!ELEMENT (col1|col2) (para+)>
```

XML fragment

```
<chapter>
<col1>
<para>The best-known...</para>
<para>The advantage...</para>
</col1>
<col2>
<para>Vestibulum erat...</para>
<para>Phasellus dapibus...</para>
</col2>
<col1>
<para>One disadvantage...</para>
<para>Thus, has...</para>
</col1>
<col2>
<para>Duis ucrsus...</para>
<para>Class aptent...</para>
</col2>
...
```

FOSI fragment

```
<e-i-c gi="col1">
<charlist inherit="1" charsubsetref="block">
<hyphen lang="en" hyph="1">
<indent inherit="1" leftind="0" rightind="*+12.5pi" firstln="*">
<algroup refpoint="first">
...
<e-i-c gi="col2">
<charlist inherit="1" charsubsetref="block">
<hyphen lang="fr" hyph="1">
<indent inherit="1" leftind="14.5pi" firstln="*">
...
<e-i-c gi="para">
<charlist inherit="1" charsubsetref="block">
<presp minimum="0.5pi" nominal="0.5pi" maximum="0.5pi" priority="med">
...
```

If the first language in the document should be output in the second column (and vice versa), `savetext`, `usetext`, and formatting pseudo-elements are needed, as demonstrated in the next figure, which uses the same DTD and XML fragments and some of the FOSI coding from **Figure 330** above.

Figure 331 Swap language columns

Vestibulum erat massa, hendrerit id, suscipit et, accumsan id, tellus. Pellentesque rutrum. Sed erat diam, imperdiet sit amet, malesuada pretium, sodales nec, Wagna. Nulla lobortis iaculis metus. Pellentesque enim. Quisque nec risus. Sed pulvinar nunc non urna. Nullam augue. Aliquam erat volutpat.

Phasellus dapibus, tellus ac feugiat aliquam, urna nibh imperdiet purus, ut eleifend elit velit in eros. Donec in augue. Vestibulum nec tortor a enim volutpat mattis. Integer pellentesque dapibus dui.

Duis ucrsus suscipit eros. Curabitur molestie iaculis nulla. Maecenas nisi justo, tempus sed, imperdiet non, tincidunt sed, lacus. Sed scelerisque lobortis sapien. In hac habitasse platea dictumst. Cras aliquet, libero vitae pellentesque ultrices, dui velit gravida massa, ut cursus tortor erat.

Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nullam dui.

The best-known dummy text is Lorem Ipsum, which is said to have originated in the 16th century. It is composed in a pseudo-Latin language which more or less corresponds to "proper" Latin. It contains a series of real Latin words.

The advantage of its Latin origin and the relative meaninglessness of is it does not attract attention to itself or distract the viewer's attention from the layout.

One disadvantage of is that in Latin certain letters appear more frequently than others. Moreover, in Latin only words at the beginning of sentences are capitalized; this means that cannot accurately represent.

Thus, has only limited suitability as a visual filler for German texts.

FOSI fragment

```
<charsubset charsubsetid="SUPPRESS"><suppress sup="1"></charsubset>
```

```
...
<e-i-c gi="col1">
<charlist inherit="1" charsubsetref="block SUPPRESS">
<savetext textid="col2.txt" conrule="#CONTENT">
...
<e-i-c gi="col2">
<charlist inherit="1" charsubsetref="block SUPPRESS">
<usetext placemnt="after"
source="<col1.fmt>,#CONTENT,</col1.fmt>,<col2.fmt>,col2.txt,</col2.fmt>">
</usetext>
...
<e-i-c gi="col1.fmt">
<charlist inherit="1" charsubsetref="block">
<hyphen lang="en" hyph="1">
<indent inherit="1" leftind="0" rightind="*+12.5pi" firstln="*">
<algroup refpoint="first">
...
<e-i-c gi="col2.fmt">
<charlist inherit="1" charsubsetref="block">
<hyphen lang="fr" hyph="1">
<indent inherit="1" leftind="14.5pi" firstln="*">
...
```

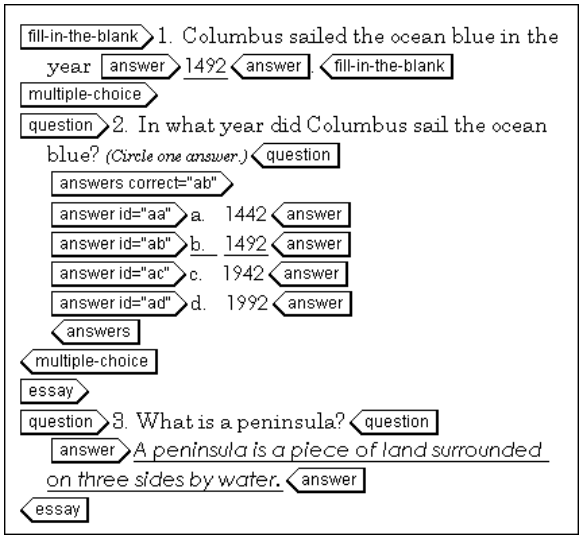
How to output student and teacher versions

Supporting a student version of a document (no answers included) and a teacher version (answers included) requires a way to indicate which version is to be output. Here, the ACL variable `$exam` is tested for student or teacher. `$exam` can be set at the command line or via a menu item.

Formatting for tests, quizzes, and the like depends on the types of questions (multiple-choice, essay, etc.) and the DTD markup. Fill-in-the-blank, multiple-choice, and essay questions are covered here. Notice the optional *answerid* attribute on <answer> and the *correct* attribute on <answers>. They identify the correct answers to multiple-choice questions in the teacher's version.

The answers in the student version cannot be seen because `highlt forpct="0"`. The length of the lines for fill-in-the-blank and essay questions is determined by the amount of text in the <answer> and the `font size` characteristic in the `answer-space charsubset`. The content of <answer> is output invisibly in the smaller size before and after the same content is output invisibly in the current `font size`. For more space, increase the `font size` characteristic in the `answer-space charsubset`. For less space, decrease the `font size` characteristic in `answer-space`.

Figure 332 Student and teacher versions (Edit window)



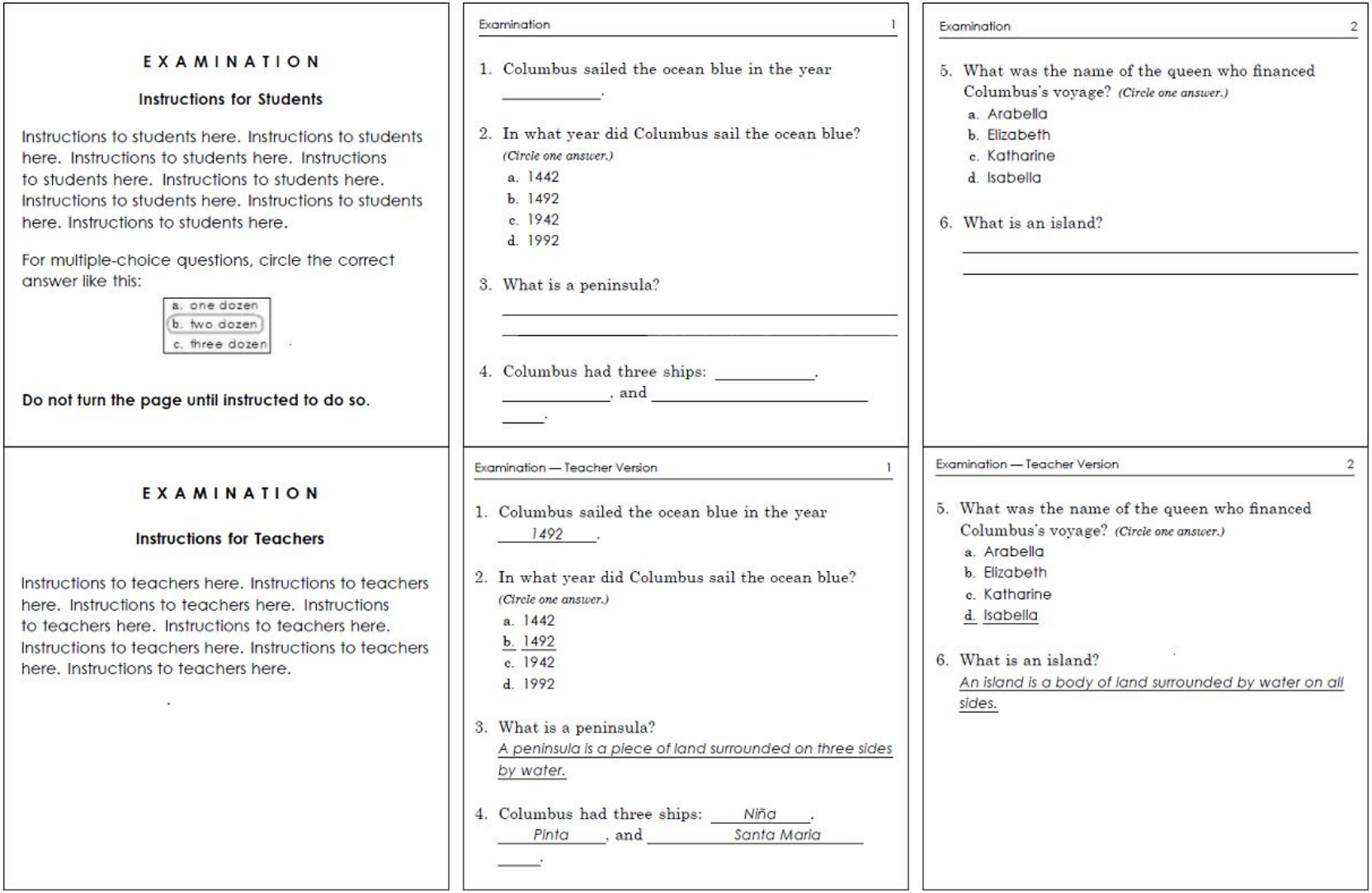
TIP

When the DTD does not provide an attribute for the FOSI to test or use, other possibilities include a namespace attribute, a pseudo-attribute, an ACL variable, or a "publishing DTD" with the necessary attribute declaration.

FOSI TIP

To ensure that content such as answers to a quiz is truly invisible, use `forpct="0"` rather than `fontclr="white"` or `fontclr="#FFFFFF"` in case a white font is visible when the document is printed on colored paper.

Figure 333 Student and teacher versions (formatted output)



EXAMINATION

Instructions for Students

Instructions to students here. Instructions to students here. Instructions to students here. Instructions to students here. Instructions to students here. Instructions to students here. Instructions to students here.

For multiple-choice questions, circle the correct answer like this:

a. one dozen
b. two dozen
c. three dozen

Do not turn the page until instructed to do so.

Examination 1

1. Columbus sailed the ocean blue in the year _____.

2. In what year did Columbus sail the ocean blue? *(Circle one answer.)*
   a. 1442
   b. 1492
   c. 1942
   d. 1992

3. What is a peninsula?
   _____
   _____

4. Columbus had three ships: _____, _____, and _____.

Examination 2

5. What was the name of the queen who financed Columbus's voyage? *(Circle one answer.)*
   a. Arabella
   b. Elizabeth
   c. Katharine
   d. Isabella

6. What is an island?
   _____
   _____

EXAMINATION

Instructions for Teachers

Instructions to teachers here. Instructions to teachers here. Instructions to teachers here. Instructions to teachers here. Instructions to teachers here. Instructions to teachers here. Instructions to teachers here.

Examination — Teacher Version 1

1. Columbus sailed the ocean blue in the year 1492.

2. In what year did Columbus sail the ocean blue? *(Circle one answer.)*
   a. 1442
   b. 1492
   c. 1942
   d. 1992

3. What is a peninsula?
   A peninsula is a piece of land surrounded on three sides by water.

4. Columbus had three ships: Niña, Pinta, and Santa Maria.

Examination — Teacher Version 2

5. What was the name of the queen who financed Columbus's voyage? *(Circle one answer.)*
   a. Arabella
   b. Elizabeth
   c. Katharine
   d. Isabella

6. What is an island?
   An island is a body of land surrounded by water on all sides.

DTD fragment

```
<!ELEMENT exam - - (fill-in-the-blank|essay|multiple-choice)+>
<!ELEMENT fill-in-the-blank - - (#PCDATA) +(answer)>
<!ELEMENT essay - - (question,answer)>
<!ELEMENT question - - (#PCDATA)>
<!ELEMENT answer - - (#PCDATA)>
<!ATTLIST answer answerid CDATA #IMPLIED>
<!ELEMENT multiple-choice - - (question,answers)+>
```

```
<!ELEMENT answers - - (answer,answer+)>
<!ATTLIST answers correct CDATA #IMPLIED>
```

XML fragment

```
<exam>
<fill-in-the-blank>Columbus sailed the ocean blue in the year
<answer>1492</answer>.</fill-in-the-blank>
<multiple-choice>
<question>In what year did Columbus sail the ocean blue?
</question>
<answers correct="ab">
<answer answerid="aa">1442</answer>
<answer answerid="ab">1492</answer>
<answer answerid="ac">1942</answer>
<answer answerid="ad">1992</answer>
</answers>
</multiple-choice>
<essay>
<question>What is a peninsula?</question>
<answer>A peninsula is a piece of land surrounded on three sides by
water.</answer>
</essay>
<fill-in-the-blank>Columbus had three ships: <answer>Niña</answer>,
<answer>Pinta</answer>, and <answer>Santa Maria</answer>.
</fill-in-the-blank>
<multiple-choice>
<question>What was the name of the queen who financed Columbus's voyage?
</question>
<answers correct="bd">
<answer answerid="ba">Arabella</answer>
<answer answerid="bb">Elizabeth</answer>
<answer answerid="bc">Katharine</answer>
<answer answerid="bd">Isabella</answer>
</answers>
</multiple-choice>
<essay>
<question>What is an island?</question>
<answer>An island is a body of land surrounded by water on all sides.
</answer>
</essay>
</exam>
```

FOSI fragment

```
<!ENTITY answer SYSTEM "circle-answer.bmp" NDATA bmp>
...
<counter initial="0" style="alphalc" enumid="answerct">
<counter initial="0" style="arabic" enumid="questionct">
<stringdecl tetxtid="correct.txt" literal="">
...
<charsubset charsubsetid="answers">
<font inherit="1" famname="Century Gothic" size="11pt">
```

```
</charsubset>
<charsubset charsubsetid="answer-lines">
<usetext source="#CONTENT" placemnt="before">
<subchars charsubsetref="answer-space">
<highlt inherit="1" scoring="1" forpct="0" scorespc="1">
...
<usetext source="#CONTENT" placemnt="after">
<subchars charsubsetref="answer-space">
<highlt inherit="1" scoring="1" forpct="0" scorespc="1">
...
<charsubset charsubsetid="answer-space">
<font inherit="1" size="0.6em" posture="upright">
<quadding inherit="1" quad="justify">
...
<e-i-c gi="answer" context="essay">
<charlist inherit="1" charsubsetref="block answers">
<font inherit="1" size="11pt" posture="italic">
<indent leftind="1.5pi" rightind="0" firstln="*">
<highlt inherit="1" scoring="1" scorespc="1">
</charlist>
<att logic="and">
<specval attname="exam" attloc="SYSTEM-VAR" attval="student">
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#NONE">
<charsubset>
<quadding inherit="1" quad="justify" lastquad="ljustify">
<highlt inherit="1" forpct="0">
<usetext source="spacefill,#CONTENT" placemnt="after">
<subchars>
<font inherit="1" size="0.5em" posture="upright">
<quadding inherit="1" quad="justify">
<highlt scoring="1" forpct="0" scorespc="1">
...
<att logic="and">
<specval attname="exam" attloc="SYSTEM-VAR" attval="teacher">
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#ANY">
<charsubset>
<font inherit="1" posture="italic">
...
<att logic="and">
<specval attname="exam" attloc="SYSTEM-VAR" attval="teacher">
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#NONE">
<charsubset>
<font inherit="1" posture="italic">
<usetext source="spacefill,#CONTENT" placemnt="after">
<subchars>
<font inherit="1" size="0.5em" posture="upright">
<quadding inherit="1" quad="justify">
<highlt scoring="1" forpct="0" scorespc="1">
...
<e-i-c gi="answer" context="fill-in-the-blank">
<charlist inherit="1" charsubsetref="inline answers"></charlist>
```

```
<att>
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#ANY">
<charsubset>
<highlt inherit="1" scoring="1" forpct="0" scorespc="1">
...
<att logic="and">
<specval attname="exam" attloc="SYSTEM-VAR" attval="student">
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#NONE">
<charsubset charsubsetref="answer-lines">
<highlt inherit="1" scoring="1" forpct="0" scorespc="1">
...
<att logic="and">
<specval attname="exam" attloc="SYSTEM-VAR" attval="teacher">
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#NONE">
<charsubset charsubsetref="answer-lines">
<font inherit="1" posture="italic">
<highlt inherit="1" scoring="1" forpct="100" scorespc="1">
...
<e-i-c gi="answer" context="answers multiple-choice">
<charlist inherit="1" charsubsetref="block answers">
<indent leftind="@+1.5pi" rightind="0" firstln="*-1.5pi">
<enumerat increm="1" enumid="answerct">
<usetext source="spacefill,answerct,\.\,0.5em,@1.5pi" placemnt="before">
<subchars>
<font inherit="1" famname="Century Schoolbook" size="11pt"
posture="upright">
...
<att logic="and">
<specval attname="exam" attloc="SYSTEM-VAR" attval="teacher">
<specval attname="answerid" attloc="answer" attval="#EQ#correct.txt">
<charsubset>
<highlt inherit="1" scoring="1" scoreoff="3pt" scorespc="1">
...
<e-i-c gi="answers" context="multiple-choice">
<charlist inherit="1" charsubsetref="block answers">
<indent leftind="1.5pi" rightind="0" firstln="*">
</charlist>
<att>
<fillval attname="correct" attloc="answers" fillcat="savetext"
fillchar="conrule">
<charsubset>
<savetext textid="correct.txt">
...
<e-i-c gi="essay" context="exam">
<charlist inherit="1" charsubsetref="block prespace keep-together">
<enumerat increm="1" enumid="questionct">
...
<e-i-c gi="exam">
<charlist inherit="1" charsubsetref="block">
<textbrk startpg="recto" pageid="doc.page" newpgmdl="global">
</charlist>
```

```
<att logic="and">
<specval attname="exam" attloc="SYSTEM-VAR" attval="student">
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="">
<charsubset>
<usetext source="<examversion.fmt>,<title.fmt>,\E X A M I N A T I O N\,
</title.fmt>,<subtitle.fmt>,\Instructions for Students\,</subtitle.fmt>,
<para.fmt>,\Instructions to students here. ...\,</para.fmt>,
<para.fmt>,\For multiple-choice questions, circle the correct answer like
this:\,</para.fmt>,</examversion.fmt>" placemnt="before">
<subchars charsubsetref="block">
...
<putgraph graphname="answer">
<subchars charsubsetref="block center prespace">
<boxing trel="top" brel="bottom" siderel="content" thick="0.5pt"
ttype="tsingle" btype="bsingle" ltype="lsingle"
rtype="rsingle">
...
<usetext source="<para.fmt>,\Do not turn the page until instructed to
do so.\,</para.fmt>">
<subchars charsubsetref="block prespace bold">
<font inherit="1" famname="Century Gothic">
...
<att logic="and">
<specval attname="exam" attloc="SYSTEM-VAR" attval="teacher">
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#NONE">
<charsubset>
<usetext source="<examversion.fmt>,<title.fmt>,\E X A M I N A T I O N\,
</title.fmt>,<subtitle.fmt>,\Instructions for Teachers\,</subtitle.fmt>,
<para.fmt>,\Instructions to teachers here. ...\,</para.fmt>,
</examversion.fmt>" placemnt="before">
<subchars charsubsetref="block">
...
<e-i-c gi="fill-in-the-blank" context="exam">
<charlist inherit="1" charsubsetref="block prespace keep-together">
<indent leftind="1.5pi" rightind="0" firstln="*-1.5pi">
<enumerat increm="1" enumid="questionct">
<usetext source="questionct,\.\,@1.5pi" placemnt="before">
...
<e-i-c gi="question" context="essay">
<charlist inherit="1" charsubsetref="block">
<indent leftind="1.5pi" rightind="0" firstln="*-1.5pi">
<usetext source="questionct,\.\,@1.5pi" placemnt="before">
...
<e-i-c gi="question" context="fill-in-the-blank">
<charlist inherit="1" charsubsetref="block"></charlist>
...
<e-i-c gi="multiple-choice" context="exam">
<charlist inherit="1" charsubsetref="block prespace keep-together">
<reset resetlist="answerct">
<enumerat increm="1" enumid="questionct">
...
```

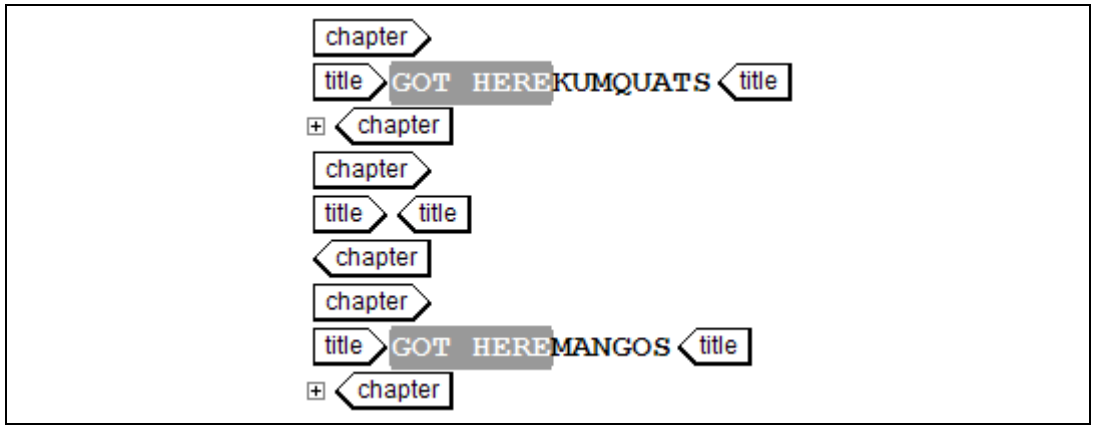
```
<e-i-c gi="question" context="multiple-choice">
<charlist inherit="1" charsubsetref="block">
<indent leftind="1.5pi" rightind="0" firstln="*-1.5pi">
<usetext source="questionct,\.\,@1.5pi" placemnt="before"></usetext>
<usetext source="\(Circle one answer.)\" placemnt="after">
<subchars>
<font inherit="1" size="9pt" posture="italic">
<keeps scope="line" keep="7">
...
<e-i-c gi="examversion.fmt">
<charlist inherit="1" charsubsetref="block">
<font inherit="1" famname="Century Gothic">
...
<e-i-c gi="para.fmt">
<charlist inherit="1" charsubsetref="block prespace">
...
<e-i-c gi="subtitle.fmt" context="examversion.fmt">
<charlist inherit="1" charsubsetref="block center prespace bold">
...
<e-i-c gi="title.fmt" context="examversion.fmt">
<charlist inherit="1" charsubsetref="block center prespace bold">
<presp condit="keep">
...
```

How to determine if an element has content

As illustrated in the following examples, there is more than one way to determine whether the current element has content. One example uses SYSTEM-FUNC and ACL, the rest use #FOSI and `specval`.

In the first example, a pseudo-element tests whether the <title> element has content. If so, "GOT HERE" is output in reverse print. However, notice that the highlighting coded in the pseudo-element applies only to the `usetext` output, not to the content of the <title> element. To apply formatting to the <title> element, see **Figure 335 #FOSI formats #CONTENT when present (Edit window view)** below.

Figure 334 #FOSI tests if element has content (Edit window view)

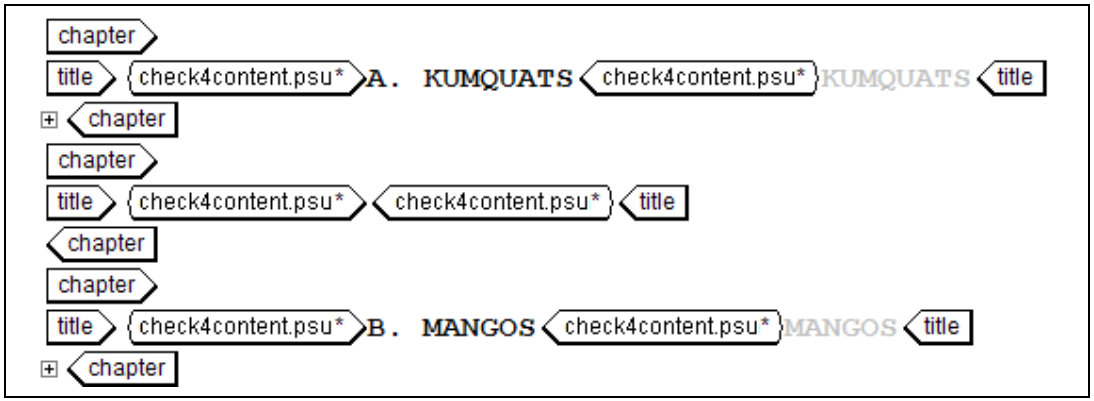


FOSI fragment

```
<e-i-c gi="title" context="chapter">
<charlist inherit="1" charsubsetref="title allcaps">
<savetext textid="chapter-title.txt" conrule="#CONTENT">
<usetext source="<check4content.psu>,</check4content.psu>"></usetext>
</charlist>
...
<e-i-c gi="check4content.psu">
<charlist inherit="1"></charlist>
<att>
<specval attname="chapter-title.txt" attloc="#FOSI" attval="#ANY">
<charsubset>
<highlt inherit="1" bckclr="#999999" fontclr="#FFFFFF">
<usetext source="\GOT HERE\"></usetext>
...
```

The next example uses the same DTD fragment and XML fragment as the previous example. In the FOSI, #CONTENT is suppressed and saved so the pseudo-element can determine whether the element has content and, if so, apply the desired formatting (in this case, chapter numbering). The suppressed content is shown in gray. Gentext tags are displayed in the graphic image to show the pseudo-element.

Figure 335 #FOSI formats #CONTENT when present (Edit window view)

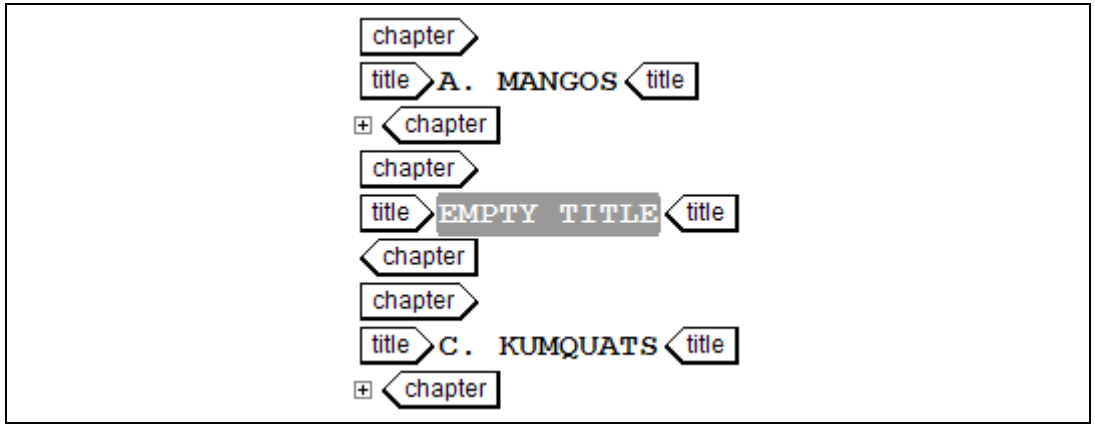


FOSI fragment

```
<counter initial="0" style="alphauc" enumid="chapterct">
<stringdecl textid="chapter-title.txt">
...
<e-i-c gi="title" context="chapter">
<charlist inherit="1" charsubsetref="title allcaps SUPPRESS">
<savetext textid="chapter-title.txt" conrule="#CONTENT">
<usetext source="<check4content.psu>,</check4content.psu>"></usetext>
...
<e-i-c gi="check4content.psu">
<charlist inherit="1"></charlist>
<att>
<specval attname="chapter-title.txt" attloc="#FOSI" attval="#ANY">
<charsubset>
<enumerat increm="1" enumid="chapterct">
<usetext source="chapterct,\. \,chapter-title.txt"></usetext>
...
```

The following example uses the same DTD and XML fragments as the previous examples, but some of the FOSI coding is different. #FOSI is used to output a message in the Edit window when the <title> element has no content. Notice that the <chapter> with the empty <title> does not have a chapter number, but the chapter is counted.

Figure 336 #FOSI displays "EMPTY TITLE" in Edit window



FOSI fragment

```
<e-i-c gi="title" context="chapter">
<charlist inherit="1" charsubsetref="title allcaps">
<savetext textid="chapter-title.txt" conrule="#CONTENT">
<usetext source="<check4content.psu>,</check4content.psu>"></usetext>
</charlist>
</e-i-c>

<e-i-c gi="check4content.psu">
<charlist inherit="1"></charlist>
<att>
<specval attname="chapter-title.txt" attloc="#FOSI" attval="#ANY">
<charsubset>
<enumerat increm="1" enumid="chapterct">
<usetext source="chapterct,\. \"></usetext>
...
<att logic="and">
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#ANY">
<specval attname="chapter-title.txt" attloc="#FOSI" attval="#NONE">
<charsubset>
<enumerat increm="1" enumid="chapterct">
<usetext source="\EMPTY TITLE\">
<subchars>
<highlt inherit="1" bckclr="#999999" fontclr="#FFFFFF">
...
```

In the last example, SYSTEM-FUNC and ACL are used to determine if an element has no content, in which case "EMPTY ELEMENT" is output.

Figure 337 SYSTEM-FUNC checks for empty element

ACL function

```
function elementcontent(window, oid)
{return oid_content(oid);}
```

FOSI fragment

```
...
<att>
<specval attname="elementcontent" attloc="SYSTEM-FUNC" attval="#NONE">
<charsubset>
<usetext source="\EMPTY ELEMENT\"></usetext>
...
```

How to force a verso page at the end of a document

Some applications require that each formatted document end with a verso page, even if it is a "blank" page (entirely blank, or with page header, footer, and/or generated text such as "This page intentionally left blank.")

One approach to supporting this in a FOSI is to use SYSTEM-FUNC at the end of the document to call a function that determines if the last page number is odd or even. However, the success of this technique depends on the document formatting, DTD structure, FOSI coding, and software version.

An alternative is to use a batch process that utilizes ACL scripting, a FOSI-generated external ASCII file, an ACL variable, and SYSTEM-VAR, as illustrated in the following figure.

NOTE: The ACL code can be entered at the command line of the document to be formatted. However, it is for proof-of-concept purposes only. It is not intended for production use. It contains dollar signs ($) with variables and omits error checking, error reporting, and comments.

The ACL code formats the document, which causes the FOSI to write the last page number to an external ASCII file (named for the `e-i-c` with `usetext userule="1"`, in this case `toptag.exp`). ACL coding opens the ASCII file, reads the number, and determines if it is odd. If so, ACL sets the variable `$addpage` to `1`, and re-formats the document. The FOSI tests the variable and forces another page.

This example shows `startpg="next"`. The page model used depends on what is coded in the current `pageset`. If a `blankpg` page model is available, it is used. Otherwise, the `versopg` page model is used, if available. If not, the `rectopg` page model is used. Or, another `pageset` can be created to handle the requirements for the blank final page.

NOTE: In the ACL fragment, `edit –current` is used to update the formatting environment so cache files are not used.

Figure 338 ACL variable tells FOSI to add another page

ACL fragment

```
$n=0;
$addpage=0;
```

```
format noprompt allpasses wait;
$d=open("toptag.exp");
$r=read($d,$n,4);
$x=chop($n,1);
if (($n % 2) == 1) {$addpage=1};
edit -current;
preview noprompt allpasses force wait;
$c=close($d);
```

FOSI fragment

```
<counter initial="0" style="arabic" enumid="folioct">
<stringdecl textid="folioct.txt" literal="">
...
<e-i-c gi="toptag">
<charlist inherit="1">
...
<usetext source="folioct.txt[BO]" placemnt="after" userule="1"></usetext>
<usetext source="<maybe-addpage.psu>,</maybe-addpage.psu>" placemnt="after">
...
<e-i-c gi="maybe-addpage.psu">
<charlist inherit="1"></charlist>
<att>
<specval attname="addpage" attloc="SYSTEM-VAR" attval="1">
<charsubset>
<usetext source="\ \">
<subchars>
<textbrk startpg="next">
...
```

How to output superscripts and subscripts

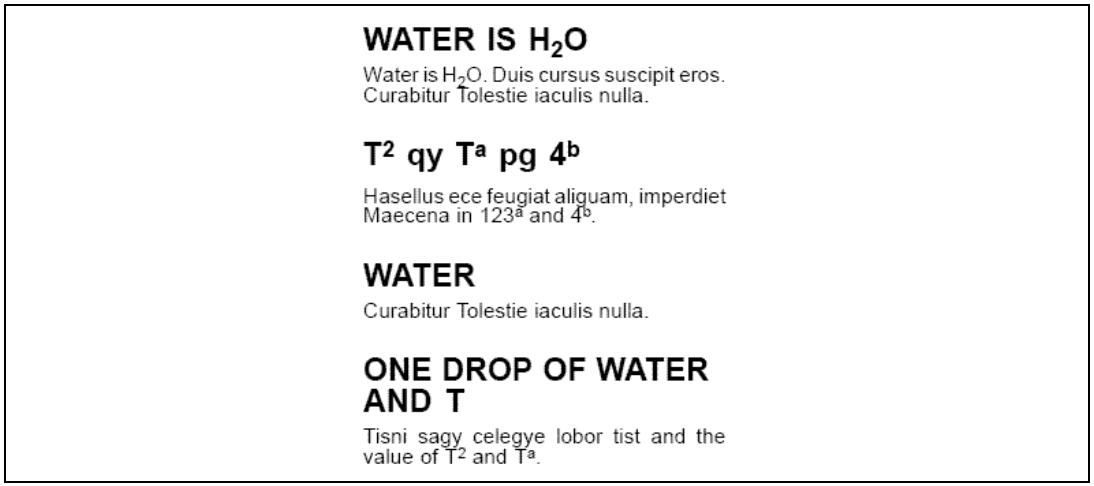
A superscript character, also known as a superior, is positioned above the baseline,. For example, in x^2, the "2" is a superscript. A subscript character, also known as an inferior, sits below the baseline. For example: H_2O. Sub- and superscripts are usually smaller than the text and are not necessarily numbers, for example: 123^a.

The `font` category's `size` and `offset` characteristics are used to support super- and subscripts. Usually, relative units, are specified so the super- and subscripts are proportional to the current size. This approach is illustrated in the following examples, which include a case in which it may be better to use fixed font sizes and offsets.

The first example illustrates the use of relative size and offset so the same `e-i-c` can be used for all contexts, regardless of their font size. In this example, `font size` in context of <title> resolves to 12.6pt, while `size` in context of <paragraph> resolves to 8.39pt.

The super- and subscripts in this example are used in text that is set "solid," that is, `font size` and `leading lead` are the same. This causes superscripts to almost touch descenders in the previous line, and subscripts to almost touch ascender characters on the next line. Additional leading is needed to avoid this, as shown in **Figure 340**.

Figure 339 Super- and subscripts in "solid" text



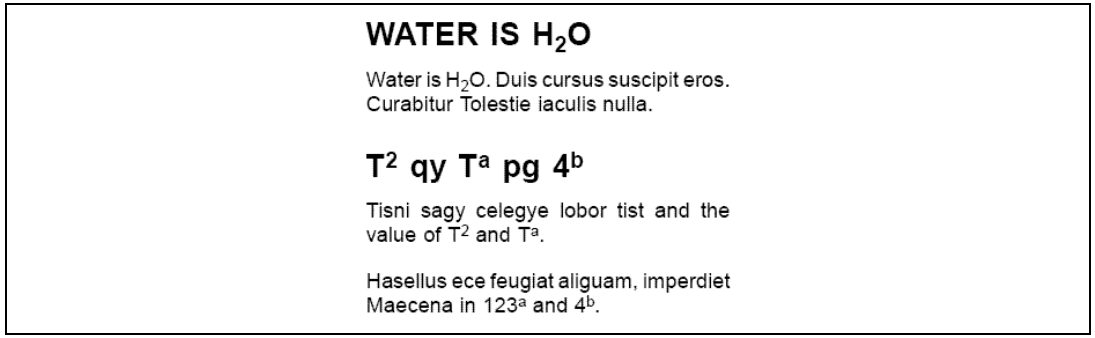
XML fragment

```
<section>
<title>WATER IS H<subscript>2</subscript>O</title>
<paragraph>Water is H<subscript>2</subscript>O. Duis...</paragraph>
</section>
<section>
<title>T<superscript>2</superscript> qy T<superscript>a</superscript>
pg 4<superscript>b</superscript></title>
<paragraph>Tisni sagy ... T<superscript>2</superscript> and
T<superscript>a</superscript>.
</paragraph>
<paragraph>Hasellus ... 123<superscript>a</superscript> and
4<superscript>b</superscript>.
...
```

FOSI fragment

```
<e-i-c gi="paragraph">
<charlist inherit="1" charsubsetref="block prespace">
<font inherit="1" size="12pt">
<leading inherit="1" lead="12pt">
...
<e-i-c gi="section">
<charlist inherit="1" charsubsetref="block">
...
<e-i-c gi="subscript">
<charlist inherit="1" charsubsetref="inline">
<font inherit="1" size="0.7em" offset="-0.3em">
...
<e-i-c gi="superscript">
<charlist inherit="1" charsubsetref="inline">
<font inherit="1" size="0.7em" offset="0.4em">
...
<e-i-c gi="title" context="section">
<charlist inherit="1" charsubsetref="title left">
<font inherit="1" size="18pt">
<leading inherit="1" lead="18pt">
...
```

Figure 340 Super- and subscripts with extra leading

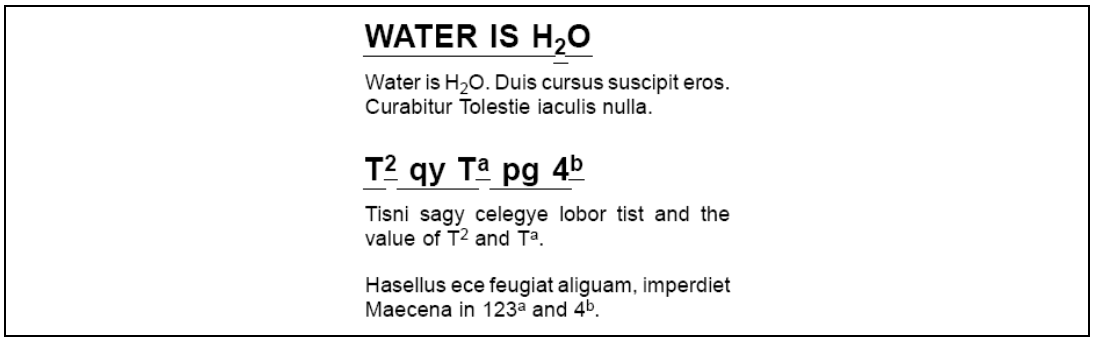


FOSI fragment

```
<e-i-c gi="paragraph">
<charlist inherit="1" charsubsetref="block prespace">
<font inherit="1" size="12pt">
<leading inherit="1" lead="14pt">...
<e-i-c gi="section">
<charlist inherit="1" charsubsetref="block">
...
<e-i-c gi="subscript">
<charlist inherit="1" charsubsetref="inline">
<font inherit="1" size="0.7em" offset="-0.3em">
...
<e-i-c gi="superscript">
<charlist inherit="1" charsubsetref="inline">
<font inherit="1" size="0.7em" offset="0.4em">
...
<e-i-c gi="title" context="section">
<charlist inherit="1" charsubsetref="title left">
<font inherit="1" size="18pt">
<leading inherit="1" lead="19pt">
...
```

The following figure illustrates the output when `highlt scoring` is used to underline text with “relative” super- and subscripts.

Figure 341 Super- and subscripts with underlining

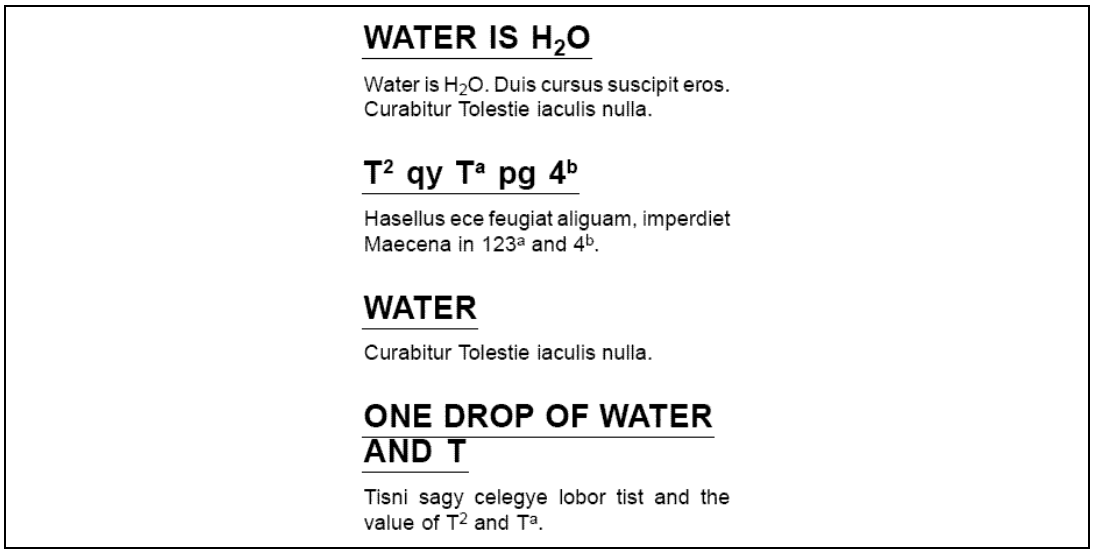


FOSI fragment

```
<e-i-c gi="title" context="section">
<charlist inherit="1" charsubsetref="title left">
<font inherit="1" size="18pt">
<leading inherit="1" lead="19pt">
<highlt inherit="1" scoring="1" scoreoff="5pt">
...
```

In the following example, super- and subscripts in context of <title> have fixed font sizes and offsets and a `highlt` category with fixed `scoreoff` values. However, as this example shows, extra leading is needed for multi-line titles.

Figure 342 Super- and subscripts in titles



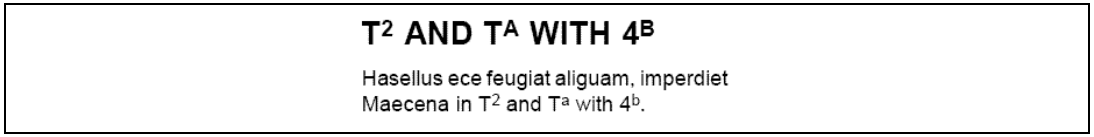
FOSI fragment

```
<e-i-c gi="subscript" context="paragraph">
<charlist inherit="1" charsubsetref="inline">
<font inherit="1" size="0.7em" offset="-0.3em">
...
<e-i-c gi="subscript" context="title">
<charlist inherit="1" charsubsetref="inline">
<font inherit="1" size="11pt" offset="-3.8pt">
<highlt inherit="1" scoreoff="2.2pt">
...
<e-i-c gi="superscript " context="paragraph">
<charlist inherit="1" charsubsetref="inline">
<font inherit="1" size="0.7em" offset="0.4em">
...
<e-i-c gi="superscript" context="title">
<charlist inherit="1" charsubsetref="inline">
<font inherit="1" size="11pt" offset="6pt">
<highlt inherit="1" scoreoff="12pt">
...
<e-i-c gi="title" context="section">
<charlist inherit="1" charsubsetref="title left">
<font inherit="1" size="18pt">
<leading inherit="1" lead="20pt">
<highlt inherit="1" scoring="1" scorewt="1pt" scoreoff="6pt">
...
```

Another approach, illustrated in the following figures, is to use `boxing` with blank left, right, and top rules. However, this underlines only the last line of a multi-line <title>. Also, as the next figure shows, when the bottom offset is relative to the last baseline, it creates the appearance of different amounts of vertical space between the <title> and <paragraph>. The figure after that shows the output with `brel="bottom"`, which in some cases results in the underline appearing to be offset too far from the line of text.

The next example shows the incorrect output that results when `e-i-cs` for super- and subscript inherit all caps from the <title> `e-i-c`.

Figure 343 Superscripts inherit all caps



XML fragment

```
<section>
<title>T<superscript>2</superscript> and T<superscript>a</superscript>
with 4<superscript>b</superscript></title>
```

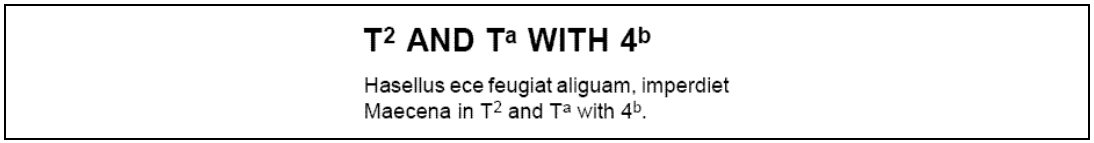
```
<paragraph>Hasellus ... T<superscript>2</superscript> and T<superscript>a
</superscript> with 4<superscript>b</superscript>.</paragraph>
...
```

FOSI fragment

```
<e-i-c gi="superscript">
<charlist inherit="1" charsubsetref="inline">
<font inherit="1" size="0.7em" offset="0.4em">
...
<e-i-c gi="title" context="section">
<charlist inherit="1" charsubsetref="title left">
<font inherit="1" size="18pt">
<leading inherit="1" lead="20pt">
<highlt inherit="1" allcap="1">
...
```

The last figure shows correct output of superscript characters.

Figure 344 Superscripts without all caps



FOSI fragment

```
<e-i-c gi="superscript">
<charlist inherit="1" charsubsetref="inline">
<font inherit="1" size="0.7em" offset="0.4em">
<highlt inherit="1" allcap="0">
...
```

How to use an attribute value for the first page number

When the page numbering in a multi-volume set continues from one volume to the next, and the volumes are in separate files that are published sequentially over a period of time, it is necessary to provide the starting page number for each volume after the first.

With Arbortext Editor, you can specify the starting page number in an attribute on the top tag in the document. This stores the page number with the document for the FOSI to use whenever the file is formatted.

NOTE: This only works on the top tag and for the first page.

This example illustrates the use of `setvalue="2"`. An attribute on the top tag in the document is used to specify the initial page number. Typically, the number is incremented in the `pageres` for the first page model used. In such cases, the attribute value should be one less than the number for the first page. In this example, the first page is numbered 101.

Figure 345 Set a new initial value for a `pagedesc counter`

XML fragment

```
<document initialize-pgno="100">
<title>Volume 2</title>
...
```

FOSI fragment

```
<counter initial="0" style="arabic" enumid="folioct">
...
<pageset id="body.page" blankpg="0" orient="portrait">
<rectopg width="6in" nomdepth="8in" xvjstretch="max">
<pageres>
<enumerat increm="1" enumid="folioct">
...
<header nomdepth="12pt" spaflow="24pt">
<usetext source="folioct">
<subchars charsubsetref="block center">
<font size="12pt">
<leading lead="12pt">
...
<e-i-c gi="document">
<charlist inherit="1">
<textbrk startpg="recto" pageid="body.page" newpgmdl="global">
</charlist>
<att>
```

```
<fillval attname="initialize-pgno" fillcat="enumerat"
fillchar="increm">
<charsubset>
<enumerat enumid="folioct" setvalue="2">
...
```

TIP

Find out the size of the paper on which the document is to be printed and the final trim size.

FOSI TIP

While working with a `region`, assign a background color (`bckclr`) to it so the `region` is visible. Use different colors for different regions on the same page so you can distinguish them. Before the FOSI is deployed, be sure to delete the color characteristics.

How to output bleed tabs

The two examples below output bleed tabs from a page `region`. However, bleed tabs can be output from `header` or `footer` as well.

Requirements for bleed tabs vary. The two examples here explore some of the possibilities.

In the first example, each section is titled with a FOSI-generated ascending uppercase letter. This letter appears white on black in a bleed tab at the right edge of each recto page in the section. When the letter is incremented for the next section, the bleed tab moves one position down the page. There is vertical space for 16 letters. The bleed tab of the seventeenth letter, "Q," is output at the top of the page in the same position as the first letter, "A."

The bleed tabs are output from a `region` that extends off the right side of the page, as **Figure 347 Preview Extended Pages display** below reveals. The `usetext` in the `region` outputs an appended string with all the section letters encountered so far. However, only the last letter appears in print/PDF output because the formatting pseudo-element that wraps each letter has two `e-i-c` in the FOSI with different occurrence specifications. `Context="notlast"` outputs a box with a white interior so the white-on-white letter does not appear. `Context="last"` outputs a box with a black interior so the white-on-black letter does appear. Note the context pseudo-element `region.ctx` that provides parental context for the occurrence settings on the `sectionct.fmt` formatting pseudo-element.

Figure 346 Bleed tabs for alphabetical sections

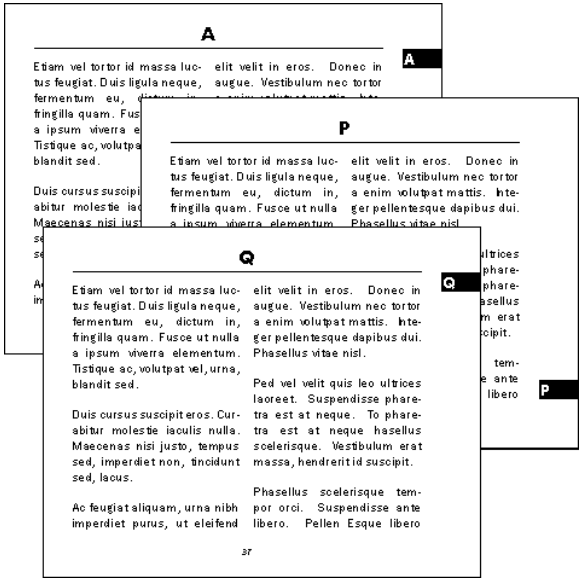
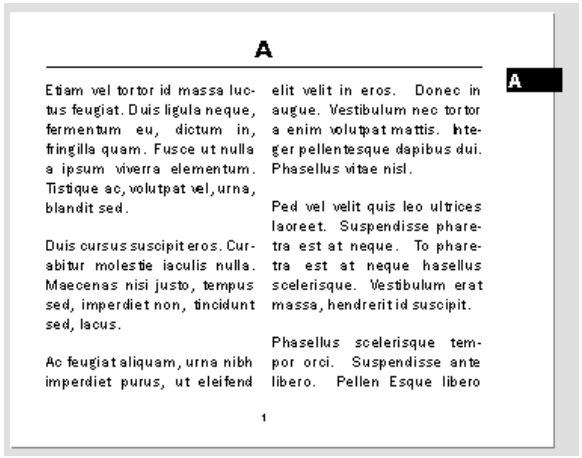


Figure 347 Preview Extended Pages display



FOSI fragment

```
<counter initial="0" style="alphauc" enumid="sectionct">
...
<stringdecl textid="sectionct.txt" literal="">
<stringdecl textid="sectionct.app" literal="">

<pageset id="body.page">
```

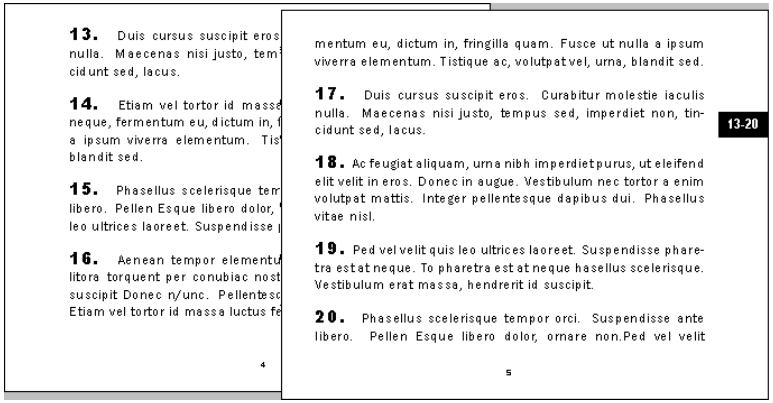
```
<rectopg width="5in" nomdepth="4in">
...
<region xoff="27.5pi" yoff="2pi" width="3pi" height="20pi" layer="-1">
<usetext source="<region.ctx>,3pt,sectionct.app[BO],</region.ctx>">
<subchars>
<font inherit="1" famname="Aharoni" size="14pt">
<highlt inherit="1" fontclr="#FFFFFF">
...
<e-i-c gi="section">
<charlist inherit="1" charsubsetref="division">
...
<enumerat increm="1" enumid="sectionct">
<savetext textid="sectionct.txt" conrule="sectionct">
...
<e-i-c gi="title" context="section">
<charlist inherit="1" charsubsetref="title center allcaps">
<usetext source="sectionct.txt" placemnt="before">
...
<att>
<specval attname="sectionct" attloc="#FOSI" attval="17">
<charsubset>
<savetext textid="sectionct.app" append="0"
conrule="<sectionct.fmt>,sectionct.txt,</sectionct.fmt>">
...
<att>
<specval attname="sectionct" attloc="#FOSI" attval="#NE#17">
<charsubset>
<savetext textid="sectionct.app" append="1"
conrule="<sectionct.fmt>,sectionct.txt,</sectionct.fmt>">
...
<e-i-c gi="region.ctx">
<charlistg inherit="1"></charlist>
</e-i-c>
<e-i-c gi="sectionct.fmt" context="region.ctx" occur="last">
<charlist inherit="1" charsubsetref="block">
<boxing toffset="12pt" boffset="3pt" loffset="0pt" roffset="0pt"
trel="first" brel="last" siderel="text" thick="1pt"
ttype="tsingle" btype="bsingle" ltype="lsingle" rtype="rsingle"
inclr="#000000" outclr="#000000">
...
<e-i-c gi="sectionct.fmt" context="region.ctx" occur="notlast">
<charlist inherit="1" charsubsetref="block">
<boxing toffset="12pt" boffset="3pt" loffset="0pt" roffset="0pt"
trel="first" brel="last" siderel="text" thick="1pt"
ttype="tsingle" btype="bsingle" ltype="lsingle" rtype="rsingle"
inclr="#FFFFFF" outclr="#FFFFFF">
...
```

In the second example, every paragraph has a FOSI-generated ascending Arabic number. The bleed tab on the right edge of each recto page contains a range of

numbers indicating the numbered paragraphs that appear on the double-page spread (verso and recto pages).

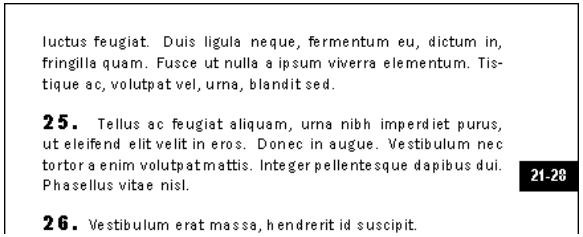
NOTE: This examples requires the use of strings modifiers, which are discussed on page 127.

Figure 348 Bleed tabs with number ranges



On the next recto page, the bleed tab moves one position down the page.

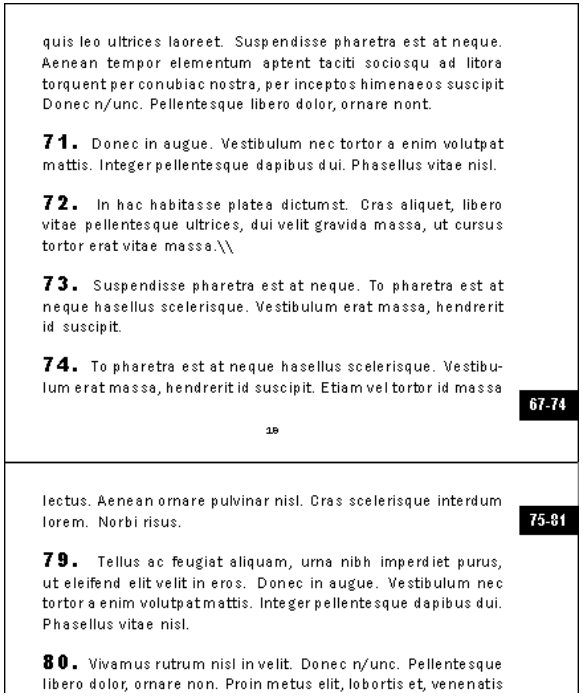
Figure 349 Bleed tabs move down the page



When a bleed tab reaches the bottom of the page, the next tab start at the top of the next page.

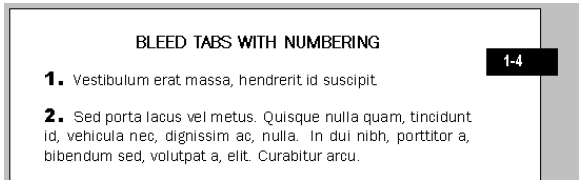
NOTE: A maximum number of cycles must be coded in the FOSI. In this example, ten tabs descend the page before returning to the top of the next page. The FOSI is coded for three cycles. After that, the bleed tabs descend off the bottom of the page.

Figure 350 Bleed tabs cycle to top of page



Preview Extended Pages shows how the tab “bleeds” off the edge of the page.

Figure 351 Preview Extended Pages display



The first page has no facing verso page, so an opening `rectopg` is used to output the first paragraph number on the recto page through the last paragraph number on the same page. The bleed tab on all other recto pages outputs the first number of the facing verso page through the last number on the recto page.

The `pageres` in the opening `rectopg` contains a `savetext` for this purpose. The `savetext` in the versopg `pageres` sets the first verso page paragraph number for the bleed tabs in the rest of the document.

The second `rectopg` utilizes a `pageref` to reference the `pagespec body.recto0`.

Figure 352 Double-page spread bleed tabs

```
<counter initial="0" style="arabic" enumid="bleedtabct">
<counter initial="0" style="arabic" enumid="paract">
<stringdecl textid="bleedtab-space.app" literal="">
<stringdecl textid="paract.txt" literal="">
<stringdecl textid="verso-first-paract.txt" literal="">
...
<pageset id="body.page">
<rectopg width="5in" nomdepth="4in">
<pageres>
<enumerat increm="1" enumid="folioct">
<savetext textid="folioct.txt" conrule="folioct">
<savetext textid="verso-first-paract.txt" conrule="paract.txt[FI]">
</pageres>
<pagespec pgid="body.recto0">
...
<region xoff="27pi" yoff="2.25pi" width="4pi" height="20pi" layer="-1">
<usetext source="bleedtab-space.app"></usetext>
<usetext source="spacefill,verso-first-paract.txt,\-\,paract.txt[BO],
spacefill,@3pi">
<subchars charsubsetref="block center bold">
<font inherit="1" famname="Arial Narrow">
<indent inherit="1" leftind="0" rightind="*+4pi" firstln="*">
<highlt inherit="1" fontclr="#FFFFFF">
<boxing toffset="12" boffset="6pt" trel="first" brel="last" siderel="text"
inclr="#000000">
<enumerat increm="1" enumid="bleedtabct">
<savetext textid="bleedtab-space.app" append="1"
conrule="<linebreak.fmt>,\ \,</linebreak.fmt>">
...
<usetext source="<check-bleedctabct.psu>,</check-bleedctabct.psu>">
...
<rectopg width="5in" nomdepth="4in">
<pageres>
<enumerat increm="1" enumid="folioct">
<savetext textid="folioct.txt" conrule="folioct">
</pageres>
<pageref pgidref="body.recto0">
</rectopg>
<versopg width="5in" nomdepth="4in" xvjstretch="max">
<pageres>
<enumerat increm="1" enumid="folioct">
<savetext textid="folioct.txt" conrule="folioct">
<savetext textid="verso-first-paract.txt" conrule="paract.txt[FI]">
</pageres>
...
<e-i-c gi="para" context="section">
```

```
<charlist inherit="1" charsubsetref="block prespace">
<enumerat increm="1" enumid="paract">
<savetext textid="paract.txt" conrule="paract" placemnt="before">
<usetext source="paract.txt,\. \" placemnt="before">
...
<e-i-c gi="check-bleedctabct.psu">
<charlist inherit="1"></charlist>
<att logic="or">
<specval attname="bleedtabct" attloc="#FOSI" attval="10">
<specval attname="bleedtabct" attloc="#FOSI" attval="20">
<specval attname="bleedtabct" attloc="#FOSI" attval="30">
<charsubset>
<savetext textid="bleedtab-space.app" conrule="\\" append="0">
...
<e-i-c gi="linebreak.fmt">
<charlist inherit="1" charsubsetref="block"></charlist>
</e-i-c>
```

How to generate change level markings and LEP

This technique for generating change level markings and a List of Effective Pages (LEP) utilizes some fancy FOSI footwork to determine the highest change level on each page for output in the footer of changed pages and in the LEP.

Every element with a change level attribute indirectly saves its change level value to a string variable. Instead of the change level number itself being saved, a pseudo-element whose name includes the change level is saved. This is done with `placemnt="before"` and `placemnt="after"` in case the element starts and ends on different pages. Lastly, the string variable is nulled out with `placemnt="after"` for the next use.

NOTE: The maximum number of change levels must be decided in order to code this technique in a FOSI.

To keep things simple, the FOSI shown in the following figure supports only three change levels, and only <paragraph> and <change> elements have *changelevel* attributes. For purposes of illustration, the value of each *changelevel* attribute is output before each element so it can easily be compared to the change level for the page.

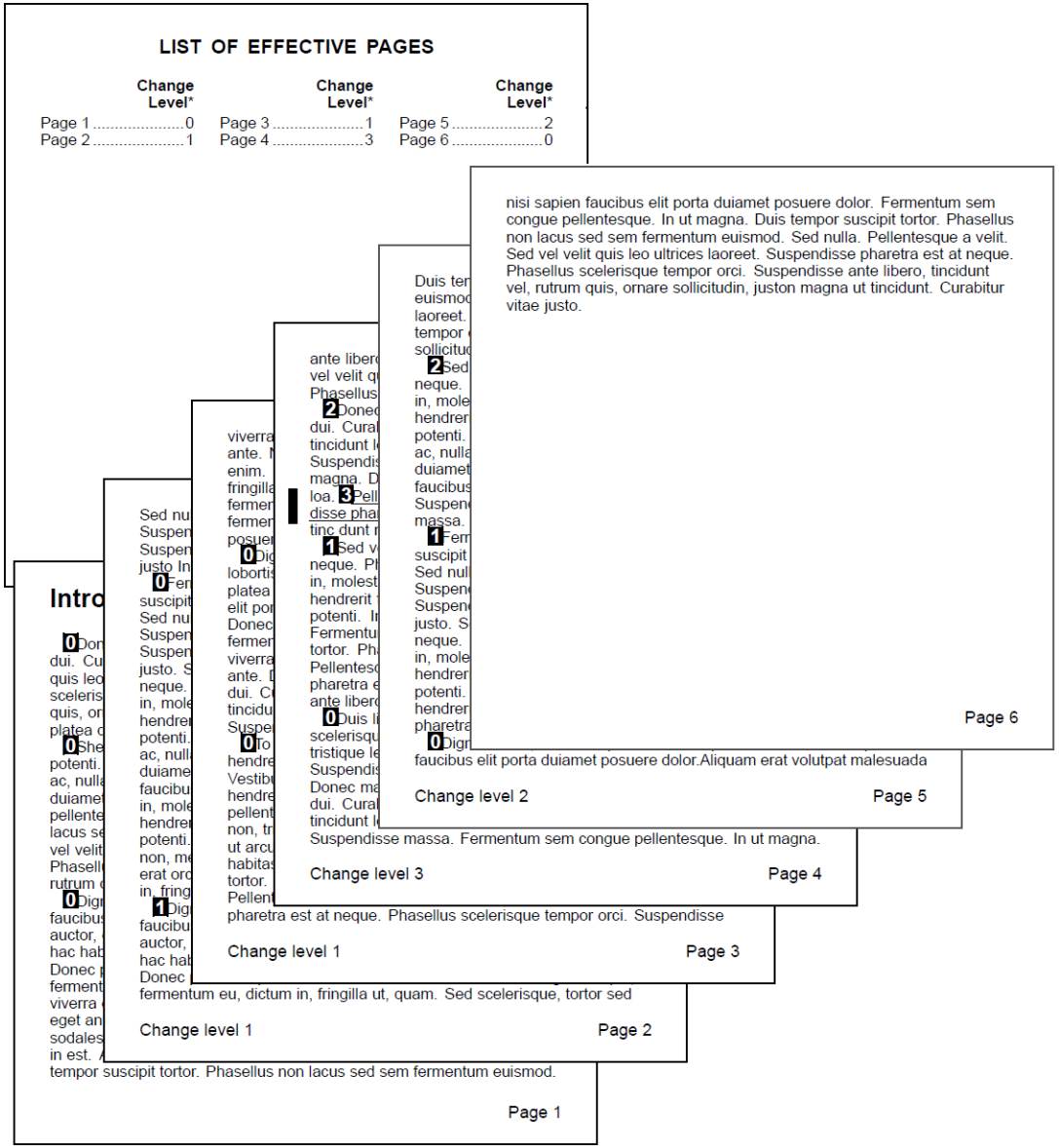
Change bars are output when the *changelevel* attribute on the current element is the same as on the top tag <doc>. In other words, change bars mark only the latest changes.

Text entities are used for attribute rules that are needed by more than one `e-i-c`.

Notice that every paragraph starts on one page and ends on the next.

NOTE: Text entities are used for FOSI code needed in more than one `e-i-c`.

Figure 353 Change level in footer and LEP



XML DTD fragment

```
<!ELEMENT doc ...>
<!ATTLIST doc changelevel NUMBER>
```

```
<!ELEMENT paragraph (#PCDATA)>
<!ATTLIST paragraph changelevel NUMBER>
<!ELEMENT change (#PCDATA)>
<!ATTLIST change changelevel NUMBER>
```

XML fragment

```
<doc changelevel="3">...
<lep>...
<chapter><title>Introduction</title>
<paragraph>...</paragraph>
<paragraph>...</paragraph>
<paragraph>...</paragraph>
<paragraph>...</paragraph>
<paragraph changelevel="1">...</paragraph>
<paragraph>...</paragraph>
<paragraph>...</paragraph>
<paragraph>...</paragraph>
<paragraph changelevel="1">...<change type="change" changelevel="2">...
</change>...</paragraph>
<paragraph changelevel="1">...</paragraph>
<paragraph>...</paragraph>
<paragraph changelevel="1">...</paragraph>
<paragraph changelevel="3">...</paragraph>
<paragraph changelevel="2">...</paragraph>
<paragraph>...</paragraph>
</chapter>
```

FOSI fragment

```
<!ENTITY changebars '<att logic="and">
<specval attname="changelevel" attloc="chgproc" attval="1">
<specval attname="changelevel" attval="1">
<charsubset>
<chgmark barthick="6pt" baroffset="1pi" type="content"></chgmark>
...
<att logic="and">
<specval attname="changelevel" attloc="chgproc" attval="2">
<specval attname="changelevel" attval="2">
<charsubset>
<chgmark barthick="6pt" baroffset="1pi" type="content"></chgmark>
...
<att logic="and">
<specval attname="changelevel" attloc="chgproc" attval="3">
<specval attname="changelevel" attval="3">
<charsubset>
<chgmark barthick="6pt" baroffset="1pi" type="content"></chgmark>
</charsubset>
</att>'>
<!ENTITY show-changelevel '
<att>
<fillval attname="changelevel" fillcat="usetext" fillchar="source">
<charsubset>
```

```
<usetext>
<subchars charsubsetref="bold">
<font inherit="1" size="1.25em">
<highlt inherit="1" fontclr="#FFFFFF">
<boxing toffset="1pt" boffset="1pt" loffset="1pt" roffset="1pt"
siderel="content" inclr="#000000"...>
...
<!ENTITY do-changelevels '
<att>
<specval attname="changelevel" attval="1">
<charsubset>
<savetext textid="do-changelevel-1.txt" placemnt="before"
conrule="<do-changelevel-1.psu>,</do-changelevel-1.psu>">
<savetext textid="do-changelevel-1.txt" placemnt="after"
conrule="<do-changelevel-1.psu>,</do-changelevel-1.psu>">
<savetext textid="do-changelevel-1.txt" placemnt="after" conrule="\\">
...
<att>
<specval attname="changelevel" attval="2">
<charsubset>
<savetext textid="do-changelevel-2.txt" placemnt="before"
conrule="<do-changelevel-2.psu>,</do-changelevel-2.psu>" >
<savetext textid="do-changelevel-2.txt" placemnt="after"
conrule="<do-changelevel-2.psu>,</do-changelevel-2.psu>">
<savetext textid="do-changelevel-2.txt" placemnt="after" conrule="\\">
...
<att>
<specval attname="changelevel" attval="3">
<charsubset>
<savetext textid="do-changelevel-3.txt" placemnt="before"
conrule="<do-changelevel-3.psu>,</do-changelevel-3.psu>">
<savetext textid="do-changelevel-3.txt" placemnt="after"
conrule="<do-changelevel-3.psu>,</do-changelevel-3.psu>">
<savetext textid="do-changelevel-3.txt" placemnt="after" conrule="\\">
</charsubset>
</att>'>
...
<stringdecl textid="doc-changelevel.txt" literal="" status="0">
<stringdecl textid="do-changelevel-1.txt" literal="" status="0">
<stringdecl textid="do-changelevel-2.txt" literal="" status="0">
<stringdecl textid="do-changelevel-3.txt" literal="" status="0">
<stringdecl textid="footer-changelevel.txt" literal="" status="0">
<stringdecl textid="lep-changelevel.txt" literal="" status="0">
<stringdecl textid="lep.app" status="1">
<floatloc floatid="lep.top" floattyp="atcolbrk" ...>
...
<pageset id="body.page">
<pageres>
<savetext textid="footer-changelevel.txt" conrule="\\">
<savetext textid="lep-changelevel.txt" conrule="\0\">
...
```

```
</pageres>
...
<footer nomdepth="12pt" spaflow="24pt">
<usetext source="do-changelevel-1.txt[FI],do-changelevel-1.txt[TO],
do-changelevel-2.txt[FI],do-changelevel-2.txt[TO],
do-changelevel-3.txt[FI],do-changelevel-3.txt[TO]">
</usetext>
<usetext source="<changelevel.psu>,</changelevel.psu>,spacefill">
<subchars charsubsetref="startline">
...
<usetext source="\Page \,folioct.txt">
<subchars charsubsetref="endline">
<savetext textid="lep.app" append="1"
conrule="<lep.fmt>,\Page \,folioct,dotfill,lep-changelevel.txt,
</lep.fmt>">
...
<pageset id="lep.page">
...
<footer nomdepth="12pt" spaflow="24pt">
<usetext source="\*0 indicates original page\,spacefill,folioct[RL]">
<subchars charsubsetref="block">
...
<flowtext numcols="3" balance="1">
<column width="9pi" topfloat="lep.top">
...
<e-i-c gi="change">
<charlist inherit="1" charsubsetref="underline">
<chgmark barthick="6pt" baroffset="1pi" type="content"></chgmark>
</charlist>
&show-changelevel;&changebars;&do-changelevels;
</e-i-c>
...
<e-i-c gi="doc">
<charlist inherit="1"></charlist>
<att>
<fillval attname="changelevel" attloc="doc"
fillcat="savetext" fillchar="conrule">
<charsubset>
<savetext textid="doc-changelevel.txt">
...
<e-i-c gi="lep">
<charlist inherit="1">
<span span="3">
<usetext source="\List of Effective Pages\">
<subchars charsubsetref="title center allcaps">
<textbrk startpg="next" pageid="lep.page" newpgmdl="local">
<span span="1">
...
<usetext source="<lep.fmt>,\Change\,</lep.fmt>,<lep.fmt>,\Level*\,
</lep.fmt>">
<subchars charsubsetref="block right bold">
```

```
<postsp minimum="2pt" nominal="2pt" maximum="2pt" priority="high">
...
<usetext source="<lep.fmt>,\Change\,</lep.fmt>,<lep.fmt>,\Level*\,
</lep.fmt>">
<subchars charsubsetref="block right bold">
<postsp minimum="2pt" nominal="2pt" maximum="2pt" priority="high">
<float flidref="lep.top" width="col" scope="lep">
...
<usetext source="lep.app">
<usetext source="\*0 indicates original page\">
<subchars charsubsetref="block keep-previous">
<font inherit="1" size="12pt">
<leading inherit="1" lead="12pt">
<presp minimum="18pt" nominal="18pt" maximum="18pt" priority="med">
<span span="1">
...
<e-i-c gi="paragraph">
<charlist inherit="1" charsubsetref="block">
<indent inherit="1" firstln="1em">
</charlist>
&show-changelevel;&changebars;&do-changelevels;
</e-i-c>
<e-i-c gi="changelevel.psu">
<charlist inherit="1">
<usetext source="footer-changelevel.txt"></usetext>
...
<e-i-c gi="do-changelevel-1.psu">
<charlist inherit="1">
<savetext textid="footer-changelevel.txt" conrule="\Change level 1\">
<savetext textid="lep-changelevel.txt" conrule="\1\">
...
<e-i-c gi="do-changelevel-2.psu">
<charlist inherit="1">
<savetext textid="footer-changelevel.txt" conrule="\Change level 2\">
<savetext textid="lep-changelevel.txt" conrule="\2\">
...
<e-i-c gi="do-changelevel-3.psu">
<charlist inherit="1">
<savetext textid="footer-changelevel.txt" conrule="\Change level 3\">
<savetext textid="lep-changelevel.txt" conrule="\3\">
...
<e-i-c gi="lep.fmt">
<charlist inherit="1" charsubsetref="block"></charlist>
</e-i-c>
```

The LEP can be exported to an ASCII file using `userule="1"` to merge adjacent pages with the same change level. In this case, pages 2 and 3 could be combined into a range (Pages 2-3) because they have the same change level (1).

NOTE: The `dotfill` in the previous example is not included in the `userule="1"` output, as shown in the first image in **Figure 354**. In such cases, formatting pseudo-elements can be included to provide the desired formatting after the exported file has been processed and formatting is restarted.

Figure 354 Exported LEPs

Exported file fragment without gentext

```
<lep.fmt>Page 1 0</lep.fmt>
<lep.fmt>Page 2 1</lep.fmt>
<lep.fmt>Page 3 1</lep.fmt>
<lep.fmt>Page 4 3</lep.fmt>
<lep.fmt>Page 5 2</lep.fmt>
<lep.fmt>Page 6 0</lep.fmt>
```

Exported file fragment with markup that generates text

```
<lep.fmt><page.fmt>Page 1 </page.fmt>
<changelevel.fmt>0</changelevel.fmt></lep.fmt>
<lep.fmt><page.fmt>Page 2 </page.fmt>
<changelevel.fmt>1</changelevel.fmt></lep.fmt>
<lep.fmt><page.fmt>Page 3 </page.fmt>
<changelevel.fmt>1</changelevel.fmt></lep.fmt>
<lep.fmt><page.fmt>Page 4 </page.fmt>
<changelevel.fmt>3</changelevel.fmt></lep.fmt>
<lep.fmt><page.fmt>Page 5 </page.fmt>
<changelevel.fmt>2</changelevel.fmt></lep.fmt>
<lep.fmt><page.fmt>Page 6 </page.fmt>
<changelevel.fmt>0</changelevel.fmt></lep.fmt>
```

FOSI fragment

```
<footer>
...
<savetext textid="lep.app" append="1"
conrule="<lep.fmt>,<page.fmt>,\Page \,folioct,</page.fmt>,
<changelevel.fmt>,lep-changelevel.txt,</changelevel.fmt>,
</lep.fmt>">
...
<e-i-c gi="changelevel.fmt" context="lep.fmt">
<charlist inherit="1" charsubsetref="endline">
...
<e-i-c gi="page.fmt" context="lep.fmt">
<charlist inherit="1" charsubsetref="startline">
<usetext source="dotfill" placemnt="after">
...
```

TIP

`Layout::apply()` also adds `atipl` tags. Please see Arbortext Editor documentation for details.

TIP

Line-numbering typically appears in the left margin, and numbering is usually reset for each page. However, <atipl:endline> could output the line numbers, if desired. Also, <atipl:startcolumn> could reset line numbering.

How to number lines of type

Line numbering is supported with Arbortext Page Layout (`atipl`), which are discussed on page 740, as follows:

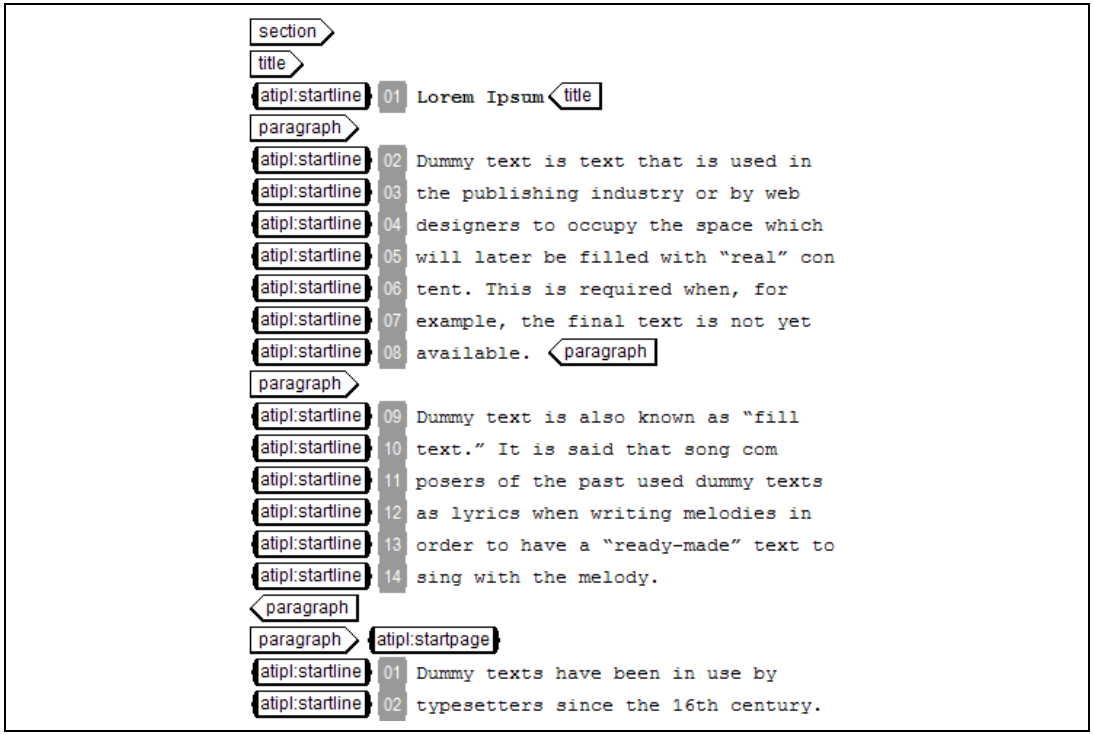
1. Code an `e-i-c` in the FOSI for <atipl:startline> to output the line numbers and another `e-i-c` for <atipl:startpage> to reset line numbers
2. Format the document with `format layout`.
3. Add page layout tags to the document with `layout::add()`.
4. Use the `set pagelayoutmarkers=on` command to display `atipl` markup in the Edit window.
5. Preview or print the document to see line numbering in formatted output.

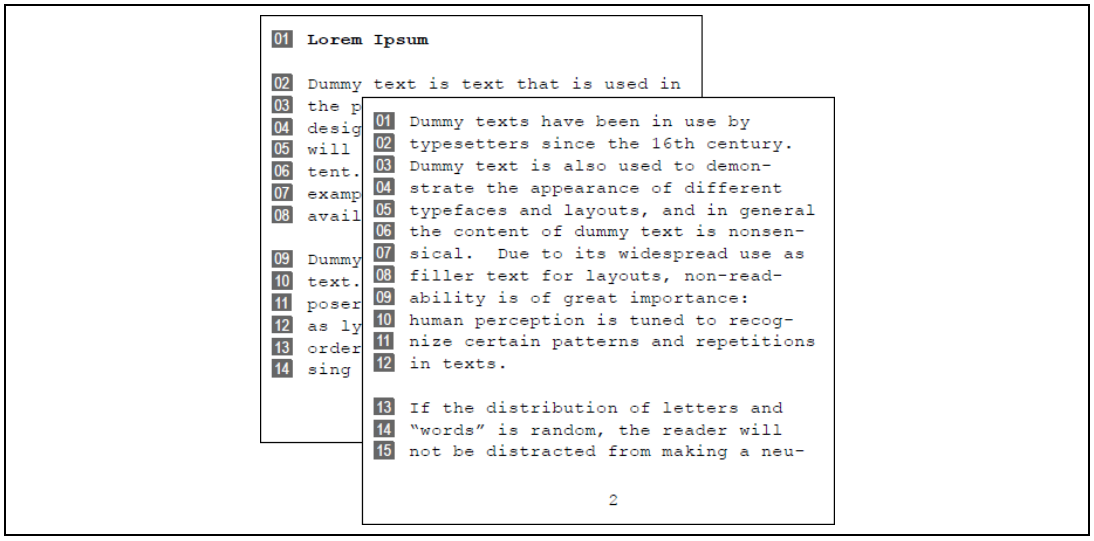
NOTE: Arbortext recommends that line numbering be used without deep content splitting enabled. Using line numbering with deep content splitting may produce unexpected results.

In **Figure 355**, the first image shows line numbers in the Edit window. The second image shows the print/PDF output. Reverse print and a different typeface are used to indicate that the numbering is not authored. Notice that FOSI code for the Edit window uses the `highlt` category for the background color while `boxing` is used for print/PDF output.

NOTE: In the image of the Edit window, the `tag display` command was used to hide the <atipl:endline> and <atipl:endpage> tags.

Figure 355 Line numbering in Edit window





TIP

If formatting seems to be "confused" or "stuck" on old formatting while you are coding a FOSI for line numbering, enter `doc_clean_cache` (page 683) at the command line to delete existing cache files.

FOSI fragment

```
<counter initial="0" style="arabic" enumid="linect" padlen="2">
...
<e-i-c gi="atipl:startpage">
<charlist inherit="1">
<reset resetlist="linect">
</charlist>
</e-i-c>
<e-i-c gi="atipl:startline">
<charlist inherit="1" charsubsetref="startline">
<enumerat increm="1" enumid="linect">
</charlist>
<att>
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#ANY">
<charsubset>
<usetext source="\ \,linect,\ \" placemnt="before">
<subchars>
<font inherit="1" famname="Arial Narrow" size="10pt" weight="medium">
<highlt bckclr="#999999" fontclr="#FFFFFF">
...
<usetext source="0.5em" placemnt="before"></usetext>
...
<att>
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#NONE">
<charsubset>
<usetext source="-1.5pi,linect" placemnt="before">
<subchars charsubsetref="bold">
<font inherit="1" famname="Arial Narrow" size="9pt" weight="medium">
<highlt fontclr="#FFFFFF">
<boxing toffset="1pt" boffset="1pt" loffset="19pt" roffset="1pt"
trel="top" brel="last" siderel="content" thick="0.5pt"
ttype="tsingle" btype="bsingle" ltype="lsingle" rtype="rsingle"
inclr="#666666" outclr="#666666">
...
<usetext source="@0pi" placemnt="before"></usetext>
...
```

How to output page layout debugging information

TIP

If you code page layout debugging capability in a FOSI, be sure to include complete information about it in the FOSI documentation.

It's good practice to include page layout debugging information in your FOSI because it often comes in handy. For instance, when there are several page models in a `pageset`, it may not be clear which is being used for each output page. Also, when optional content does not appear in page headers/footers, there is no way to tell if the header/footer is correctly formatted.

With page layout debugging code in the FOSI, each page model can output a string that identifies it, and hard-coded string variables can be output in headers/footers in order to verify their formatting.

An ACL variable is used to turn the debugging information on and off. It can be entered at the command line, or a menu item or keymapping could be used instead. To avoid accidentally publishing pages with debugging information, a message can be displayed in the Edit window to alert the user that debugging mode is enabled.

In this example, FOSI code is shown for one page model. In practice, the code needs to be included for all page models. In addition, the `savetexts` that provide values for the debugging strings must be coded for every element that could be the top tag in a document.

Figure 356 Conditional debugging strings

ACL command

```
set $debug=yes
```

FOSI fragment

```
<stringdecl textid="debug-doc-security.txt" literal="">
<stringdecl textid="debug-recto0.txt" literal="">
<stringdecl textid="debug-recto1.txt" literal="">
<stringdecl textid="debug-rectobb.txt" literal="">
<stringdecl textid="debug-versobf.txt" literal="">
<stringdecl textid="debug-verso1.txt" literal="">
<stringdecl textid="debug-wp-security.txt" literal="">
<stringdecl textid="doc-security.txt" literal="">
<stringdecl textid="wp-security.txt" literal="">
...
<!-- Include the following code for each page model -->
<!-- $debug=yes shows debugging text-->
<rectopg>

<pagespec pgid="wp-recto0">
```

```
<header>
<usetext source="doc-security.txt[BO],debug-doc-security.txt[FI]">
<subchars charsubsetref="doc-security-header">
...
<usetext source="wp-security.txt[BO],debug-wp-security.txt[FI]">
<subchars charsubsetref="wp-security-header">
...
<region xoff="36pt" yoff="36pt" width="1in" height="36pt">
<usetext source="debug-wp-recto0.txt[FI]"></usetext>
</region>
...
<!-- Include the following code in the e-i-c for any element -->
<!-- that can be the top tag in a document -->
<e-i-c gi="toptag">
<charlist inherit="1"...>
...
<!-- debug strings appear in preview/print/PDF when $debug=yes -->
<att>
<specval attname="debug" attloc="SYSTEM-VAR" attval="yes">
<charsubset>
<savetext textid="debug-doc-security.txt" conrule="\doc-security\">">
<savetext textid="debug-wp-security.txt" conrule="\wp-security\">g
<savetext textid="debug-recto0.txt" conrule="\recto0\">
<savetext textid="debug-recto1.txt" conrule="\rectot1\">
<savetext textid="debug-rectobb.txt" conrule="\rectobb\">
<savetext textid="debug-versobf.txt" conrule="\versobf\">
<savetext textid="debug-verso1.txt" conrule="\verso1\">
</charsubset>
</att>
<!-- message appears in Edit window when $debug=yes -->
<att logic="and">
<specval attname="debug" attloc="SYSTEM-VAR" attval="yes">
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="yes">
<charsubset>
<usetext source="\NOTE: Debugging strings appear in page headers/footers\">
<subchars charsubsetref="onscreen-message">
...
```

How to create modular FOSIs

Many organizations strive to standardize formatting and corporate identity image for their publications. However, the variety of documents generally requires more than one DTD. Modular FOSIs are used to standardize formatting for documents that follow the same or different DTDs.

When formatting should be identical for the same element in different documents and/or DTDs, FOSI modules can be created for common formatting and used in different FOSIs to easily standardize formatting. When the formatting changes, only one module must be modified to affect all relevant documents. When standardized formatting must be supported, this approach makes stylesheet maintenance much faster.

A modular FOSI references a chunk of FOSI code (module) that is also referenced by other modular FOSIs. Each module is a single source of standardized formatting. A change to the module affects every FOSI that references it.

For example, the formatting for legal information may be the same for many different kinds of documents. A FOSI module is created that contains the formatting specific to the markup for legal information. The FOSIs for all documents that use that formatting are then coded to reference that module. If the formatting changes in the future and the module is changed, all FOSIs that reference the modified module are updated when they are compiled.

This section describes how to create a modular stylesheet, which includes documenting how each module is used, and employing naming conventions that facilitate file maintenance.

NOTE: A module is technically an SGML file entity that is declared in the `.fos` file. It can be edited like any `.fos`, `.sgm`, or `.xml` file.

Construct a modular FOSI as follows:

1. Review each FOSI and create charsubsets and/or pseudo-elements, as appropriate, for any common code within a FOSI.
2. Determine the shared formatting and the FOSIs that share it.
3. Code `rsrcdesc`, `secdesc`, `pagedesc`, `styldesc`, and `ftndesc` directly in the `.fos` files. Do not include them in modules. This enables multiple modules to be referenced in each “desc.” Also, categories can be coded directly in the `.fos` file intermixed with module references.

4. Develop a naming convention for FOSIs and modules that make their purpose obvious.
5. Create as many file entities as needed for the following FOSI components and name them appropriately.

   - `Hyphrule`
   - `Charfill`
   - `Counter`
   - `Stringdecl`
   - `Floatloc`
   - `Pageset`
   - `Sectoken`
   - `Charsubset`
   - `Docdesc`
   - `Envdesc`
   - `e-i-c`
   - `Ftnatts`

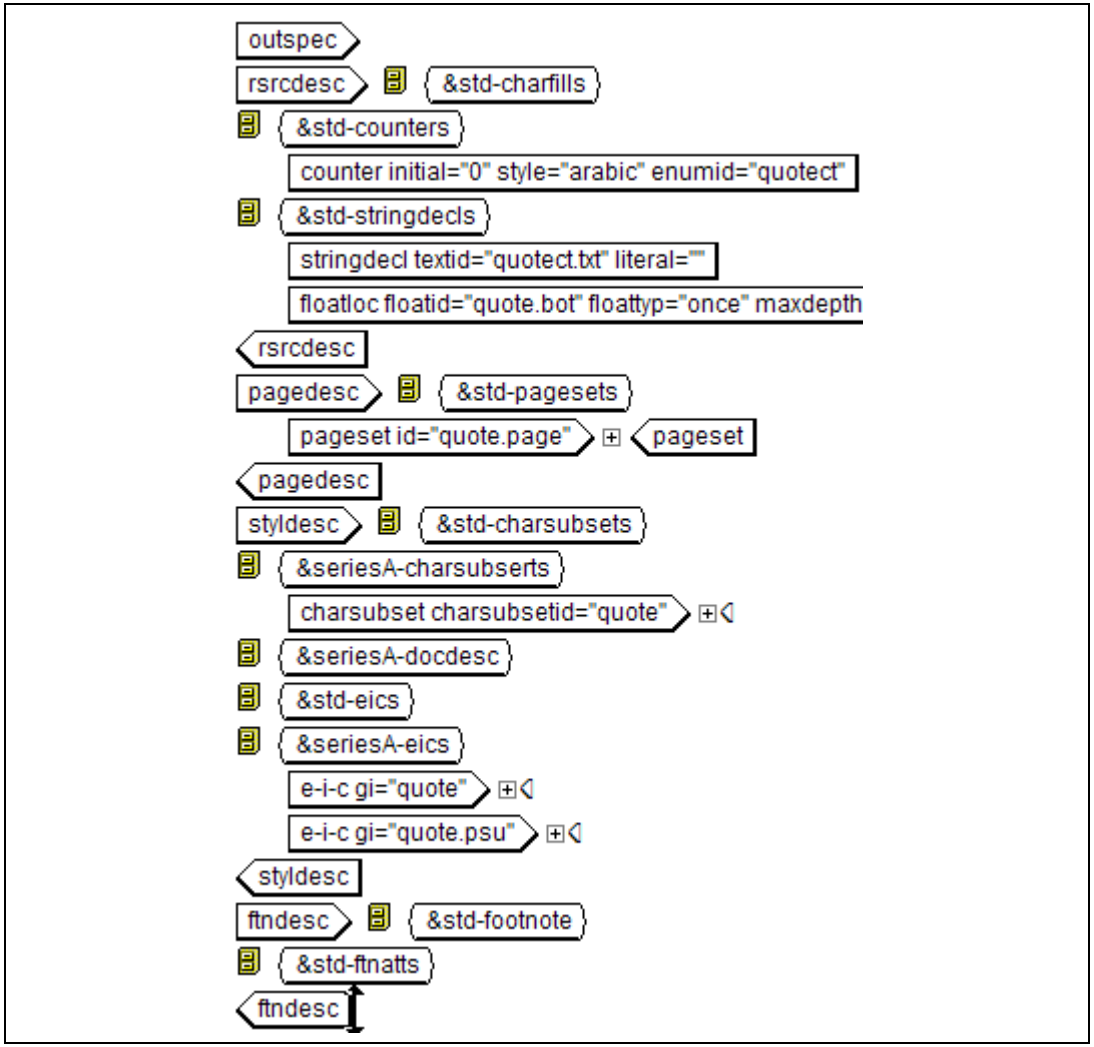
6. Code FOSI-specific code directly in each `.fos` file.
7. Insert the file entities into the appropriate `.fos` files.
8. Document the entities referenced by each FOSI. **Table 82 FOSIs and referenced entities** on page 626 provides an example.

   NOTE: Be sure to update the documentation whenever you modify a module and delete or add references to a module. Up-to-date documentation is the key to successful modular FOSIs.

The image below shows SGML file entities displayed in the tagged FOSI editor. The naming conventions for the modules reflect their purpose. In this example, the FOSI references “standard” modules that are referenced by all FOSI. The FOSI also references “Series A” modules, which are referenced by all publications in Series A. In addition, FOSI categories are coded directly in the `.fos` file.

The FOSI fragment that follows the image shows the file entity declarations in the `.fos` file.

Figure 357 Modular FOSI with SGML file entities (Edit window view)



FOSI fragment

```
<!ENTITY std-charfills SYSTEM "std-charfills.ent">
<!ENTITY std-counters SYSTEM "std-counters.ent">
<!ENTITY std-stringdecls SYSTEM "std-stringdecls.ent">
<!ENTITY std-pagesets SYSTEM "std-pagesets.ent">
<!ENTITY std-charsubsets SYSTEM "std-charsubsets.ent">
<!ENTITY seriesA-charsubserts SYSTEM "seriesA-charsubsets.ent">
<!ENTITY seriesA-docdesc SYSTEM "seriesA-docdesc.ent">
<!ENTITY std-eics SYSTEM "std-eics.ent">
```

```
<!ENTITY seriesA-eics SYSTEM "seriesA-eics.ent">
<!ENTITY std-footnote SYSTEM "std-footnote.ent">
<!ENTITY std-ftnatts SYSTEM "std-ftnatts.ent">
```

The following table shows an example of documenting entities referenced by each FOSI.

Table 82 FOSIs and referenced entities

	Entities				
Series A FOSIs	**std-eics.ent**	**seriesA-eics.ent**	**std-footnote.ent**	**std-ftnatts.ent**	
quote.fos	✓	✓			
reference.fos	✓	✓	✓	✓	

How to support "lights out" formatting

"Lights out" refers to unattended formatting for print/PDF output. The idea is that human intervention should be necessary only when a formatting error condition is reported. Reporting of error conditions is controlled by formatting fault settings, which are described on page 664.

Some formatting requirements rule out a "light out" approach. For example, when hyphenation is used, a review of the hyphenation breaks may be needed to find and fix bad breaks. Hyphenation of people's names is particularly problematic.

Some formatting requirements pose a challenge for a "lights out" approach. ACL and other scripting may be needed. For example, business rules may require a <warning> to fit in one column. One way to handle this is to develop scripting for a pre-print process that formats each <warning> to determine if it is too deep. If so, ACL can set an attribute on <warning> to tell the FOSI to use a smaller font size, leading, vertical spacing, etc. ACL then formats the <warning> again. The process is repeated until the <warning> fits or the smallest acceptable font size is reached, in which case the FOSI displays a message in the Edit window that the <warning> is too long and must be edited.

Another example is when element content is used for `header` or `footer`. The space allocated for page headers and footers is fixed. If the content is too deep, the header/footer oversets the area and may intrude into the Flowing Text Area. One way to prevent header/footer oversets is to add a *short-version* attribute to the element whose content is output in the header/footer, and code the FOSI to use the attribute value if it exists. The ACL `formaterrorhook` (page 831) is used in a pre-print process to intercept `hdrftroverset` formatting faults and display a dialog for the user to enter a shorter title. The shorter title is then stored in the *short-version* attribute, and the document is formatted again.

Following are some guidelines for formatting that is amenable to the "lights out" approach.

- Set formatting fault thresholds to report oversets, missing graphics, and other faults, as appropriate.
- Use line and page formatting that requires no review. For example:
  - Do not use hyphenation.
  - Use ragged right rather than justification so too much or too little word spacing is not an issue.

- Use a ragged bottom page layout (no vertical justification) so too little or too much vertical space is not an issue.
- Avoid two-column layouts with Z-paging and page-wide figures and tables that do not float. Such layouts can result in an ambiguous reading order that confuses readers, as illustrated on page 850. Instead, use L-paging to eliminate the ambiguity.

- Use prioritized `keeps` settings to manage line-, column-, and page-breaking.
- Decrease cell margins to help prevent oversets in tables.
- Provide a draft print capability for hardcopy review which helps reveal formatting problems ahead of time.
  - For multi-column layouts, print just one column to leave space for handwritten edits.
  - Include a PROOF/DRAFT overlay/underlay.
  - Include a date stamp.