Pseudo-attributes

Arbortext Editor supports a processing instruction that serves as a *pseudo-attribute*. Its purpose is to extend a DTD that cannot be changed. When one or more pseudo-attributes is defined for a doctype, they are available for every element in the DTD.

DOCTYPE TIP
Which to use: ACL variable or pseudo-attribute? An ACL variable is not saved with the document; a pseudo-attribute setting is.

A pseudo-attribute is defined in a file named `atiattr.cf` located in the same directory as the `.dtd` file. If the `atiattr.cf` file is deleted from the doctype directory, pseudo-attribute settings in documents are not recognized.

The file `atiattr.cf` must include the following, where `attributename` indicates the name of the attribute you are creating. More than one `attributename` may be included.

atiattr.cf file

```
tag: _xattrs single NULL NULL
sty: attributename NULL NOTEX TEXT NULL NULL NULL NULL NULL
attr: ATTR_ATI
```

NOTE: The doctype must be recompiled after creating `atiattr.cf`.

A FOSI treats pseudo-attributes the same as DTD attributes. The value of a pseudo-attribute can be tested with the `specval` category or used with the `fillval` category.

Pseudo-attributes-related ACL

Pseudo-attributes do not appear in the Modify Attributes dialog and are not shown in tags in the Edit window. A value is assigned to a pseudo-attribute with the `modify_tag` command entered at the command line or issued via a custom menu item or keymapping.

Entering `mt attributename=""` sets the pseudo-attribute to an empty string, which is written to file as <?Pub Lcl attributename="">.

Pseudo-attribute PIs can be deleted using the `write -nopi` command. Pseudo-attribute PIs can be manipulated using ACL functions such as `oid_delete_attr` and `oid_modify_attr`. However, Find▸Processing Instruction does not locate pseudo-attribute PIs.

Pseudo-attribute example

In the first example, a pseudo-attribute named *edition* is set to `largeprint` on the top tag in the document, <book>. When the document is formatted, different formatting specs are used.

Figure 405 Pseudo-attribute changes formatting specs

`Atiattr.cf` file

```
tag: _xattrs single NULL NULL
sty: edition NULL NOTEX TEXT NULL NULL NULL NULL NULL
attr: ATTR_ATI
```

XML fragment

```
<book><?Pub Lcl edition="largeprint">
...
```

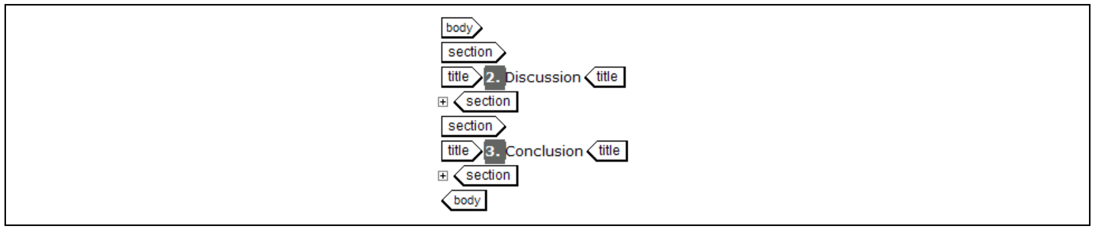
FOSI fragment

```
<e-i-c gi="book">
<charlist inherit="1">
<font inherit="1" size="10pt" ...>
<leading inherit="1" lead="11pt">
<span span="2">
<textbrk startpg="recto" pageid="regularprint.page" newpgmdl="global">
</charlist>
<att>
<specval attname="edition" attval="largeprint">
<charsubset>
<font inherit="1" size="14pt" ...>
<leading inherit="1" lead="17pt">
<span span="1">
<textbrk startpg="recto" pageid="largeprint.page" newpgmdl="global">
...
```

A pseudo-attribute can be used to generate the correct numbering when a document is incomplete. In the next example, the image shows the Edit window display after the *sectionnumber* pseudo-attribute has been set for the first <setion> tag in the file. The generated numbering that results is highlighted in the image. Notice that the <section> tag does not indicate an attribute has been set.

The purpose of the unconditional <att> is to save the `counter` value set tin the `charlist` or set by the `fillval` in the preceding `att`. Notice the `enumerat` category in the FOSI fragment specifies `setvalue="1"` to set the `counter` to the value specified in the `increm` characteristic.

Figure 406 Pseudo-attribute for starting numbers (Edit window view)



`Atiattr.cf` file

```
tag: _xattrs single NULL NULL
sty: sectionnumber NULL NOTEX TEXT NULL NULL NULL NULL NULL
attr: ATTR_ATI
```

XML fragment

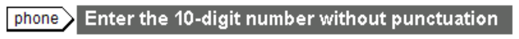
```
<body>
<section><?Pub Lcl sectionnumber="2">
<title>Discussion</title>
</section>
<section><?Pub Lcl sectionnumber="3">
<title>Conclusion</title>
...
```

FOSI fragment

```
<counter enumid="sectionct" initial="0" style="arabic">
...
<e-i-c gi="section" context="body">
<charlist inherit="1" charsubsetref="block">
<enumerat increm="1" enumid="sectionct">
</charlist>
<att>
<fillval attname="sectionnumber" attloc="section" fillcat="enumerat"
fillchar="increm">
<charsubset>
<enumerat enumid="sectionct" setvalue="1">
...
<att>
<charsubset>
<savetext textid="sectionct.txt" conrule="sectionct,\. \">
...
<e-i-c gi="title" context="section">
<charlist inherit="1" charsubsetref="block">
<usetext source="sectionct.txt" placemnt="before">
...
```

The following example shows setting a pseudo-attribute to cause authoring guidelines to be displayed in the Edit window. In this case, the pseudo-attribute is set on the top tag, which can be used to toggle the display of all guidelines.

Figure 407 Authoring/editing instructions in the Edit window



`Atiattr.cf` file

```
tag: _xattrs single NULL NULL
sty: guidelines NULL NOTEX TEXT NULL NULL NULL NULL NULL
attr: ATTR_ATI
```

XML fragment

```
<book><?Pub Lcl guidelines="yes">
...
<phone></phone>
...
```

FOSI fragment

```
...<e-i-c gi="phone">
<charlist inherit="1" charsubsetref="inline">
...
<att logic="and">
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#ANY">
<specval attname="guidelines" attloc="book" attval="yes">
<charsubset>
<usetext source="\ Enter the 10-digit number without punctuation \">
<subchars charsubsetref="authoring-guidelines">
...
```

Processing instruction-related ACL

This section introduces some ACL relating to PIs. Please refer to Arbortext Editor documentation for more information.

- The `oid_name` function returns the name that Arbortext Editor uses for a PI.
- The `write` command has options related to PIs.
  - `-pi` writes Arbortext Editor PIs.
  - `-nopi` removes all Arbortext Editor PIs from a document, subject to the `set writechangetracking` command. If `set writechangetracking` is not set, the `write` command writes out the document with all pending changes highlighted.
- The settings for the `set writechangetracking` command specify the output from the `write` command as follows:
  - `Original` shows the document as if all pending changes have been rejected
  - `Changesapplied`, shows the document as if all pending changes have been accepted
  - `Changeshighlighted` (the default) shows all pending change records
- The `set writepi` command determines how the `save`, `write`, and `save_as` commands treat Arbortext Editor PIs. The settings are:
  - `On`, which saves Arbortext Editor PIs.
  - `Off`, which removes Arbortext Editor PIs.
  - `Default`, which saves Arbortext Editor PIs except for <_display>.

  The scope of the setting is global. The default setting is `writepi=default`.

  NOTE: When `-pi` or `-nopi` is specified for the `write` or `save` commands, that value overrides the `set writepi` setting.