

Pointing, Linking, and Stand Off Markup

James Cummings

January 2010



Linking, segmentation and alignment

In some texts we need to be able

- to link disparate elements without using the `@xml:id` attribute;
- to *segment* text into elements and to mark arbitrary points within documents
- to represent **correspondence** or **alignment** among groups of text elements
- to **synchronize** elements of a text, representing temporal correspondences and alignments among text elements
- to specify that one text element is *identical* to or a *copy* of another
- to *aggregate* possibly noncontiguous elements
- to specify that different elements are *alternatives* to one another and to express *preferences* among the alternatives
- to store markup separately from the the data it describes

Underlying assumptions

- Use W3C identifying, pointing and linking mechanisms where possible
- Use *@xml:id* to identify an element directly
- Use XPointer to point to elements that do not have an *@xml:id*

Complex pointing

The standard URI scheme allows for pointing

- to documents other than the current document
- to a particular element in a document other than the current document using its `xml:id`;

but we also need to point

- to a particular element using its position in the XML element tree (standard XPointer schemes)
- at arbitrary content in any XML document using TEI-defined XPointer schemes

Some W3C XPointer schemes (1)

- element** Identify elements by position within parent, recursively.
- left** Locates the point immediately preceding its argument. The argument may return a node, node set, range, or point. (TEI Submitted)
- range** Locates a range between two points in an XML information set. Takes two pointer arguments which locate the boundaries of the range by two points, and are interpreted as fragment identifiers. (TEI Submitted)

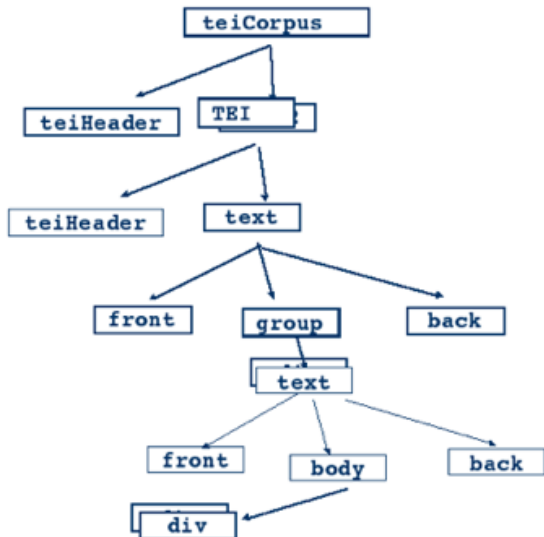
Some W3C XPointer schemes (2)

- right** Locates the point immediately following its argument. The sole argument is a pointer, which is treated as if it were a fragment identifier itself. (TEI Submitted)
- string-range** Locates a range based on character positions. Takes three arguments: a pointer, an offset, and a length. (TEI Submitted)
- xmlns** Bind a prefix for use in subsequent pointer parts e.g. `xmlns(xs=http://www.w3.org/2001/XMLSchema)`

Some W3C XPointer schemes (3)

- xpath1** Locates a node or node set within an XML Information Set. The single argument is an XPath path as defined in the W3C XPath 1 Recommendation.
- xpath2** Locates a node or node set within an XML Information Set. The single argument is an XPath path as defined in the W3C XPath 2 Recommendation.
- xpointer** The rich scheme including XPaths and ranges described in the XPointer Recommendation

Test document for XPointer schemes




```
<text>
  <body>
    <!-- seven divs here -->
    <div xml:id="lastterm">
      <p>... </p>
      <p>... </p>
      <p>
        <emph>'But'</emph>, said
        <name key="Stalky">Stalky</name>,
        'come to think of it, we've done more giddy
        jesting with the Sixth since we've been
        passed over than any one else in the last
        seven years.' </p>
      </div>
    </body>
  </text>
```

Examples for XPointer schemes

- element():

```
<ptr target="stalky.xml#element(lastterm)"/>  
<ptr target="stalky.xml#element(1/1/8)"/>
```

- xpointer() and xmlns():

```
<ptr  
  target="stalky.xml#xmlns(t=http://www.tei-c.org/ns/1.0)  
  xpointer(/t:TEI/t:text/t:body/t:div[8])"/>
```

- range():

```
<ptr
  target="stalky.xml#xmlns(t=http://www.tei-c.org/ns/1.0)
  range(xpath1(/t:TEI/t:text/t:body/t:div[8]/t:p[3]),
  xpath1(/t:TEI/t:text/t:body/t:div[8]/t:p[5]))"/>
```

```
<ptr
  target="stalky.xml#xmlns(t=http://www.tei-c.org/ns/1.0)
  xpointer(/t:TEI/t:text//t:p[t:name[.='Stalky']])"/>
```

Note that the last expression returns multiple nodes.

A daily use for XPointer

The W3C XInclude specification is a good way to write composite documents; the `<include>` element's *@href* attribute allows for XPointers:

```
<div> <xi:include href="stalky.xml#  
xmlns(t=http://www.tei-c.org/ns/1.0)  
xpointer(/t:TEI/t:text//t:p[t:name[.='Beetle']])"/>  
</div>
```

Generic linking



The core TEI `<ptr>` and `<ref>` elements let you do the point to point linking we are used to on web pages, relying on XML IDs for internal links:

```
<p>Wikipedia has a a good starter page on  
<ref  
  target="http://en.wikipedia.org/wiki/Maneki_Neko">waving  
cats</ref>, with links to more esoteric  
resources; our own pictures are in  
section <ref target="#cats">3</ref>  
</p>
```

'linking' Module

The **linking** module adds `<link>` to let you specify a point to point relationship between two or more elements:

```
<p xml:id="beetle1">You're a despondin' brute, Beetle</p>  
<p xml:id="beetle2">An' who the dooce is this  
Raymond Martin, M. P.?' demanded Beetle</p>  
<link targets="#beetle1 #beetle2"/>
```

Note that this is establishing a connection, not a direction.

Groups of links

`<linkGrp>` is provided to group together sets of `<link>`s. In the following example, it allows for stand-off notes, and characterisation of those notes:

```
<l xml:id="l2.79">A place there is, betwixt earth,  
air and seas</l>  
<l xml:id="l2.80">Where from Ambrosia, Jove  
retires for ease.</l>  
<l xml:id="l2.88">Sign'd with that Ichor which from Gods  
distills.</l>  
<note xml:id="n2.79">  
  <bibl>Ovid Met. 12.</bibl>  
  <quote xml:lang="la">  
    <l>Orbe locus media est, inter terrasq; fretumq;</l>  
    <l>Cælestesq; plagas </l>  
  </quote>  
</note>  
<note xml:id="n2.88">Alludes to <bibl>Homer, Iliad 5</bibl>  
</note>  
<linkGrp type="imitationnotes">  
  <link targets="#n2.79 #l2.79"/>  
  <link targets="#n2.88 #l2.88"/>  
</linkGrp>
```

Segmenting text, and marking arbitrary points within documents

This module adds three useful new elements:

- `<ab>` marks a block of text with no special semantic interpretation
- `<seg>` marks a range of text with no special semantic interpretation
- `<anchor>` marks an arbitrary point in the text

The first two have helpful *@type* and *@subtype* attributes.

Marking points

`<anchor >` is comparable to HTML anchors:

`<p>`He was merely working up to a peroration, and the boys knew it; but McTurk cut through the frothing sentence, the others echoing: `</p>`

`<p>`' `<anchor xml:id="MTa"/>`I appeal to the Head, sir.' `</p>`

`<p>`' `<anchor xml:id="Ba"/>`I appeal to the head, sir.' `</p>`

`<p>`' `<anchor xml:id="Sa"/>`I appeal to the Head, sir.' `</p>`

`<p>`It was their unquestioned right. Drunkenness meant expulsion after a public flogging. They had been accused of it. The case was the Head's, and the Head's alone. `</p>`

Anonymous blocks

In this inscription, there are separate lines, but they are not poetry, or paragraphs, so we isolate them with `<ab>` (or if we didn't want to encapsulate them we could use `<lb/>`):

```
<div>  
  <ab>JOSEPH STORY</ab>  
  <ab>ONLY SON OF</ab>  
  <ab>WILLILAM W. AND EMELYN STORY</ab>  
  <ab>BORN MAY 3rd 1847</ab>  
  <ab>AT BOSTON U. S. A</ab>  
  <ab>DIED NOV. 23rd 1853</ab>  
  <ab>AT ROME</ab>  
</div>
```

Segments

There are more specific elements elsewhere in the TEI for marking sentences, words and characters, but sometimes we need to mark an arbitrary span, using `<seg>`:

```
<q>Don't say <q>  
  <seg type="stutter">I-I-I</seg>'m afraid, </q>  
Melvin, just say <q>I'm afraid. </q>  
</q>
```

Correspondence and alignment

First, consider the representation of a manuscript page:



Manuscript Text

```
<ab xml:id="N6">  
  <lb/>and hat hire  
don in obedience ðe cnoweð hire manere  
<lb/>and hire strençõe. he mai ðe vttre  
riwle chaungen efter <lb/>wisdom also he  
isihð te inre mai beon best iholden.  
<anchor xml:id="N_6"/>  
  <lb/>Non ancre bi  
mine rede ne schal makien professiun.  
<lb/>pet is. bihoten ase hest.  
</ab>
```

Correspondence and alignment (cont.)

Now lets look at an edited version and a translation:

```
<p xml:id="edited_6">Nan ancre bi mi read ne schal  
makien professiun—pet is, bihaten  
ase heast—bute preo þinges,  
pet beoð obedience, chastete, ant  
stude—steaðeluestnesse  
<!-- ... -->  
</p>
```

```
<p xml:id="translated_6">My advice  
is that no anchoress should make  
profession—that is, bind herself to  
a vow—of more than three things,  
which are obedience, chastity, and  
stability of abode  
<!-- ... -->  
</p>
```

Correspondence and alignment (cont.)

We can express a relationship between the texts as follows:

```
<linkGrp type="translations">
  <link targets="#edited_6 #translated_6"/>
<!-- ... -->
</linkGrp>
<linkGrp type="editions">
  <link targets="#N-f2r #N6"/>
<!-- ... -->
</linkGrp>
```

meaning ‘this paragraph in the translated edition corresponds to text at that anchor in the original’.

There are many other ways of dealing with material like this!

Synchronizing time-based material

If you are linking together sequences which are aligned by time, there is a special stand-off linking element `<when>`, grouped inside a `<timeline>`. It has attributes:

absolute an absolute time for the event

interval the length of the gap since the last event

unit the unit of time in which the interval value is expressed

since a link to the previous event

```
<timeline xml:id="t1" origin="#w0" unit="ms">
  <when xml:id="w0" absolute="11:30:00"/>
  <when xml:id="w1" interval="unknown" since="#w0"/>
  <when xml:id="w2" interval="100" since="#w1"/>
  <when xml:id="w3" interval="200" since="#w2"/>
  <when xml:id="w4" interval="150" since="#w3"/>
</timeline>
```

These when objects can be used in a `<link>` to relate time events to points in the text.

Aggregating non-contiguous elements

The `<join>` element is used like `<link>`, pointing to 2 or more identified fragments of text. It claims that they could be joined to create a new virtual element (the *@result* attribute). `<joinGroup>` is provided to aggregate `<join>`s.

<join> example source

```
<lg>
  <l>
    <seg xml:id="L1">E</seg>lizabeth it is in vain you say</l>
  <l>"<seg xml:id="L2">L</seg>ove not" – thou sayest it in so sweet a way:</l>
  <l>
    <seg xml:id="L3">I</seg>n vain those words from thee or L. E. L.</l>
  <l>
    <seg xml:id="L4">Z</seg>antippe's talents had enforced so well:</l>
  <l>
    <seg xml:id="L5">A</seg>h! if that language from thy heart arise,</l>
  <l>
    <seg xml:id="L6">B</seg>reath it less gently forth – and veil thine eyes.</l>
  <l>
    <seg xml:id="L7">E</seg>ndymion, recollect, when Luna tried</l>
  <l>
    <seg xml:id="L8">T</seg>o cure his love – was cured of all beside –</l>
  <l>
    <seg xml:id="L9">H</seg>is follie – pride – and passion – for he died.</l>
</lg>
```

<join> example

```
<p>
  <join
    targets="#L1 #L2 #L3 #L4 #L5 #L6 #L7 #L8 #L9" result="name">
    <desc>The beloved's name</desc>
  </join>
</p>
```

(from Edgar Allan Poe).

Elements as alternatives to one another

The `<alt>` element is used to indicate that two elements are mutually exclusive. `<altGroup>` is provided to aggregate `<alt>`s.

Example: the first time we transcribed this text, we saw

```
<ab>WILLILAM W. AND EMELYN STORY</ab>
```

but on another look it says

```
<ab>WILLIAM W. AND EMELYN STORY</ab>
```

Can this be a genuine change since our first visit? or just a mistake?
Let's keep both:

```
<ab xml:id="W1">WILLILAM W. AND EMELYN STORY</ab>  
<ab xml:id="W2">WILLIAM W. AND EMELYN STORY</ab>  
<alt mode="excl" targets="#W1 #W2"/>
```

`@weights` and `@mode` assign weight to the judgement, and allow for relationships other than mutually-exclusive.

Another way to express alternation

The global *@exclude* attribute can be used by any element to indicate another element to which it is allergic:

```
<ab exclude="#W4" xml:id="W3">WILLIAM W. AND EMELYN STORY</ab>  
<ab exclude="#W3" xml:id="W4">WILLIAM W. AND EMELYN STORY</ab>
```

Conclusions

The linking module provides a wide range of tools to let you describe relationships between parts of your text. If you need these, remember:

- You should work out a naming scheme to assign ID attributes. You will need a lot of them
- There are often several ways to do things; use the more specialized markup when you can to make it easier for others to read. Don't rely on *@type* attributes with undefined meanings everywhere
- Control your vocabulary for token attributes like *@type*
- The TEI only takes you as far as *markup*. Implementing all this to make a fancy interactive text exploration web site may be a lot of work.