Non-Standard Characters and Glyphs

TEI@Oxford

December 2010

Non-standard characters

In the majority of cases, Unicode already copes for most characters that scholars need. However, there are some medieval abbreviation signs and other uncommon glyphs which have yet to make it into Unicode. Moreover, sometimes you may wish to record palaeographic variants in a single character in order to facillitate scribal identification (double-compartment versus single-comparment 'a' for example).

Since this is uncommon in the majority of text encoding, but a necessary component for those who need it, the TEI deals with this by having another optional module 'Gaiji'.

Characters in TEI: Unicode

- Unicode is the only supported character encoding schema. This means that entities for characters are deprecated, and the recommended daily use is for UTF-8 encoded text, as in

```
<persName   xml:lang="is">Katrín Þórdís Matthíasdóttir</persName>
<placeName   xml:lang="sr-Cyrl">Црна Гора</placeName>
<persName   xml:lang="el-grc">Φλ. Θάλλος</persName>
```

- There is a clean mechanism to use non-Unicode characters
- all appropriate text content models are set to allow a mixture of CDATA and <g> (where <g> is a reference to a non-Unicode character)
- all elements have an optional attribute *@xml:lang* to record the language used
- there are no places where an attribute is used to hold pure text

Non-Unicode characters

If you wish to encode characters or specific glyphs which do not appear in Unicode:

- define them in a series of <charDesc> elements, inside <encodingDesc> in the TEI header
- refer to them using using the <g> element in the body of your text

Inside <charDesc>:

- the <char> element defines a character which is not available in the current document character set
- the <glyph> element annotates an existing character (usually providing a glyph that shows how a character appeared in the original document)

'Gaiji' module elements

- The TEI Gaiji module includes:
  - <charDecl> (declaration about nonstandard characters and glyphs)
  - <char> (descriptive information about a character)
  - <charName> (name of the character)
  - <charProp> (some property of the character)
  - <g> (represents a non-standard character or glyph)
  - <glyph> (descriptive information about a glyph)
  - <glyphName> (name of the glyph)
  - <localName> (locally defined name for a property)
  - <unicodeName> (name of a registered Unicode normative property)
  - <value> (single value for some property or attribute)
  - <mapping> (one or more characters related to the current character or glyph)

  You can also use <graphic> to provide a picture of your character or glyph.

Defining a character

A new character can assigned to a position in the Unicode Private Use Area (PUA), and also described in terms of Unicode combining characters:

```
<charDecl>
  <char  xml:id="ydotacute">
    <charName>LATIN SMALL LETTER Y WITH DOT ABOVE AND ACUTE</charName>
    <charProp>
      <localName>entity</localName>
      <value>ydotacute</value>
    </charProp>
    <mapping  type="composed">#x0079;#x0307;#x0301;</mapping>
    <mapping  type="PUA">U+E0A4</mapping>
  </char>
</charDecl>
```

One can then reference the character declaration by pointing to its *@xml:id* attribute with a *@ref* attribute on the <g> element:

```
<g  ref="#ydotacute"/>
```

Defining a local glyph

A new glyph variant can also be assigned to a position in the Unicode Private Use Area (PUA) and provide standardized form as a fallback:

```
<charDecl>
  <glyph  xml:id="z103">
    <glyphName>LATIN LETTER Z WITH TWO STROKES</glyphName>
    <mapping  type="standardized">Z</mapping>
    <mapping  type="PUA">U+E304</mapping>
  </glyph>
</charDecl>
```

This can now be referenced in the text using the <g> element:

```
<g  ref="#z103"/>
```

Another <charDecl> Example

```
<charDecl>
  <glyph  xml:id="r1">
    <glyphName>LATIN SMALL LETTER R WITH ONE FUNNY STROKE</glyphName>
    <charProp>
      <localName>entity</localName>
      <value>r1</value>
    </charProp>
    <graphic  url="r1img.png"/>
  </glyph>
  <glyph  xml:id="r2">
    <glyphName>LATIN SMALL LETTER R WITH TWO FUNNY STROKES</glyphName>
    <charProp>
      <localName>entity</localName>
      <value>r2</value>
    </charProp>
    <graphic  url="r2img.png"/>
  </glyph>
</charDecl>
```

Referencing Character Declarations

One references the character declarations and glyphs by pointing to their *@xml:id* attribute with a *@ref* attribute on the <g> element:

```
<p>Wo<g   ref="#r1">r</g>ds in this
manusc<g   ref="#r2">r</g>ipt are sometimes
written in a funny way.</p>
```

More uses of <g>

It is also possible to override what appears in the text by using markup like this

```
<g   ref="#z103">z</g>
```

The content of the <g> element can be used immediately without any lookup, or we may recover a graphic or other mapping by following the link to z103.

Using <g> for variant letter forms

In facsimile-level (i.e. strictly diplomatic) transcriptions it is customary to preserve a number of variant letter forms (dotless i, tall s, round d and so on) which are otherwise generally normalised. Here too <g> may be used, even where the variant forms are defined in Unicode, allowing one to record the presence of the variant form of the letter but have the option of displaying it as an ordinary letter.

In the following example the Icelandic word *leiða* (to lead) is written with a dotless i and round d:

```
le<g ref="#iinodot"/>
<g ref="#drot"/>a
```

Definitions for dotless u and rounded d

```
<charDecl>
  <glyph   xml:id="inodot">
    <glyphName>LATIN SMALL LETTER DOTLESS I</glyphName>
    <mapping   type="dipl">i</mapping>
    <mapping   type="facs">U+0131</mapping>
  </glyph>
</charDecl>
<charDecl>
  <glyph   xml:id="drot">
    <glyphName>LATIN SMALL LETTER D ROTUNDA</glyphName>
    <mapping   type="dipl">d</mapping>
    <mapping   type="facs">U+F109</mapping>
  </glyph>
</charDecl>
```

Note that these are glyphs, rather than characters.

PUA: Private Use Area

The developers of the Unicode Standard have set aside an area of the codespace for the private use of software vendors, user groups, or individuals. As of this writing (Unicode 5.0), there are around 137,000 code points available in this area, which should be enough for most needs. No code point assignments will be made to this area by standard bodies and only some very basic default properties have been assigned (which may be overwritten where necessary by the mechanism outlined in this chapter). Therefore, unlike all other code points defined by the Unicode Standard, PUA code points should not be used directly in documents intended for blind interchange.