1 ODD SUBSET

1.1 Representation of non-standard characters and glyphs

Despite the availability of Unicode, text encoders still sometimes find that the published repertoire of available characters and glyphs is inadequate to their needs. This is particularly the case when dealing with ancient languages, for which encoding standards do not yet exist. These Guidelines provide a mechanism to satisfy that need, while retaining compatibility with standards.

1.1.1 Is your journey really necessary?

When encoders encounter some graphical unit in a document which is to be represented electronically, the first issue to be resolved should be 'Is this really a character?' To determine whether a particular graphical unit *is* a character or not, see «D4-42».

If the unit is indeed determined to be a character, the next question should be 'Has this character been encoded already?' In order to determine whether a character has been encoded, encoders should follow the following steps:

1. Check the Unicode website, in particular the page "Where is my Character?", and the associated character code charts. Alternatively, users can check the latest published version of The Unicode Standard , though the website is often more up to date than the printed version, and should be checked for preference.

   The pictures ('glyphs') in the Unicode code charts are only meant to be representative, not definitive. If a specific form of an already encoded character is required for a project, refer to the guidelines contained below under Annotating Characters.

2. Check the Proposed New Characters webpage (`http://unicode.org/alloc/Pipeline.html`) to see whether the character is in line for approval.

3. Ask on the Unicode email list to determine if a proposal is pending, or whether this is indeed a new character (or if this not a character at all, in which case it would not be eligible for addition to the Unicode Standard).

Since there are now close to 100,000 characters in Unicode, chances are good that what you need is already there, but it might not be easy to find, since it might have a different name in Unicode. Look again, this time at other sites, for example `http://www.eki.ee/letter`, which also provide searches based on scripts and languages. Take care, however, that all the properties of what seems to be a relevant character are consistent with those of the character you are looking for. For example, if your character is definitely a digit, but the properties of the best match you can find for it say that it is a letter, you may have a character not yet defined in Unicode.

In general, it is advisable to avoid those Unicode characters which are presentation forms, letter-like symbols, or number forms. However, if the character you are looking for is being used in a notation (rather than as part of the orthography of a language) then it is quite acceptable to select characters from the Mathematical Operators block, provided that they have the appropriate properties (i.e. `So`: Symbol, Other; or `Sm`: Symbol, Math).

An encoded character may be precomposed or it may be formed from base characters and combining diacritical marks. Either will suffice for a character to be "found" as an encoded character.

If there are several possible Unicode characters to choose amongst, it is good practice to consult other colleagues and practitioners to see whether a consensus has emerged in favour of one or other of them.

If, however, no suitable form of your character seems to exist, the next question will be: 'Does the graphical unit in question represent a variant form of a known character, or does it represent a completely unencoded character?' If the character is determined to be missing from the Unicode Standard, it would be helpful to submit the new character for inclusion (see `http://unicode.org/pending/proposals.html`).

These guidelines will help you proceed once you have identified a given graphical unit as either a variant or an unencoded character. Determining this will require knowledge of the contents of the document that you have. The first case will be called *annotation* of a character, while the second case will be called *adding* of a new character. How to handle graphical units that represent variants will be discussed below (1.1.3. Annotating characters) while the problem of representing new characters will be dealt with in section 1.1.4. Adding new characters.

While there is some overlap between these requirements, distinct specialized markup constructs have been created for each of these cases as explained in section 1.1.2. Markup constructs for representing non-standard characters below. The following section will then proceed to discuss how to apply them to the problems at hand, discussing annotation of existing characters in section 1.1.3. Annotating characters and finally creation of new ones in 1.1.4. Adding new characters.

1.1.2 Markup constructs for representing non-standard characters

The gaiji module provides a mechanism to declare characters additional to those available from the document character set. XML allows for a document (or document component) to declare its encoding, thus restricting the characters that can be encoded directly within it without using numeric character references. For example, an XML document which begins `<?xml version="1.0" encoding="iso-8859-1"?>` can include non-ISO-8859-1 characters only by representing them as numeric character references. In such a case, it might be convenient to declare as additional characters some characters already defined by the Unicode Standard. Generally speaking, however, the document character set will be Unicode, and this mechanism will be needed only for characters not defined by the Unicode Standard.

The mechanism provided here consists functionally of two parts:

- an element `<g>`, which serves as a proxy for new characters or glyphs
- an element `<charDesc>`, which serves as a container for information about this character. Such information is needed to identify and process this character properly.

When the gaiji module is included in a schema, the `<charDesc>` element is added to the tei.encoding class, and the `<g>` element is added to the phrase class. These elements and their components are documented in the rest of this section.

The Unicode standard defines properties for all the characters it defines in the Unicode Character Database, knowledge of which is usually built into text processing systems. Since the characters that are instantiated with the `<g>` element either do not exist in Unicode at all, or are not available in the subset of Unicode defined by the XML declaration currently in use, their properties are not readily available. The `<charDesc>` element makes it possible to store such properties for use by applications in a standard way.

The list of attributes (properties) for characters is modelled on those in the Unicode Character Database, which distinguishes normative and informative character properties. Additional, non-Unicode, properties may also be supplied. Since the list of properties will vary with different versions of the Unicode Standard, there may not be an exact correspondence between them and the list of properties defined in these Guidelines.

Usage examples for these elements are given below at 1.1.3. Annotating characters and 1.1.4. Adding new characters. The gaiji module itself is formally defined in section 1.1.6. Additional module defined by this section below. It defines the following additional elements:

- **<charDesc>** provides descriptive information about characters or glyphs additional to that implied by the current document character set.
  - No attributes other than those globally available (see definition for tei.global.attributes)
- **<g>** represents a non-standard character or glyph. Selected attributes:

  ref points to a description of the character or glyph intended.

The following elements may appear within a `<charDesc>` element:

- **<desc>** contains a brief description of the purpose and application for an element, attribute, or attribute value.
  - No attributes other than those globally available (see definition for tei.global.attributes)
- **<char>** provides descriptive information about a character which is not otherwise available in the document character set.
  - No attributes other than those globally available (see definition for tei.global.attributes)
- **<glyph>** provides descriptive information about a character glyph which is not otherwise available in the document character set.
  - No attributes other than those globally available (see definition for tei.global.attributes)

The `<char>` and `<glyph>` elements have similar contents and are used in similar ways, but their functions are different. The `<char>` element is provided to define a character which is not available in the current document character set, for whatever reason, as stated above. The `<glyph>` element is used to annotate an existing character, usually by providing a specific glyph that shows how a character appeared in the original document. Unicode codepoints refer to a very general abstract character, which can be rendered by a very large number of possible representations. The `<glyph>` element is provided for cases where the encoder wants to specify a specific glyph (or family of glyphs) out of all possible glyphs.

The Unicode Standard recommends naming conventions which must be followed strictly where the intention is to annotate an existing Unicode character, and which should also be used as a model when creating new names for characters or glyphs. For convenience of processing, the following distinct elements are proposed for naming characters and glyphs:

- **<charName>** contains the name of a character, expressed following Unicode conventions.
  - No attributes other than those globally available (see definition for tei.global.attributes)
- **<glyphName>** contains the name of a glyph, expressed following Unicode conventions for character names.
  - No attributes other than those globally available (see definition for tei.global.attributes)

Within both `<char>` and `<glyph>`, the following elements are available:

- **<gloss>** identifies a phrase or word used to provide a gloss or definition for some other word or phrase.

  target identifies the associated term element
- **<charProp>** provides a name and value for some property of the parent character or glyph.
  - No attributes other than those globally available (see definition for tei.global.attributes)
- **<desc>** contains a brief description of the purpose and application for an element, attribute, or attribute value.
  - No attributes other than those globally available (see definition for tei.global.attributes)
- **<mapping>** contains one or more characters which are related to the parent character or glyph in some respect, as specified by the type attribute.
  - No attributes other than those globally available (see definition for tei.global.attributes)
- **<figure>** indicates the location of a graphic, illustration, or figure.

  entity names the external entity within which the graphic image of the figure is stored.

  url The location of the graphical image as a URL

  width The display width of the image

  height The display height of the image

  scale The scale factor to the apply to the image to make it the desired display size
- **<remarks>** contains any commentary or discussion about the usage of an element, attribute, class, or entity not otherwise documented within the containing element.
  - No attributes other than those globally available (see definition for tei.global.attributes)

Four of these elements (`<gloss>`, `<desc>`, `<figure>` and `<remarks>`) are defined by other TEI modules, and their usage here is no different from their usage elsewhere. The `<figure>` element, however, is used here only to link to an image of the character or glyph under discussion, or to contain a representation of it in SVG. Several `<figure>` elements may be given, for example to provide images with different resolution, or in different formats. The *type* attribute may be used to specify the type of image.

The `<mapping>` element is similar to the standard TEI `<equiv>` element. While the latter is used to express correspondence relationships between TEI concepts or elements and those in other systems or ontologies, the former is used to express any kind of relationship between the character or glyph under discussion and characters or glyphs defined elsewhere. It may contain any Unicode character, or a `<g>` element linked to some other `<char>` or `<glyph>` element, if, for example, the intention is to express an association between two non-standard characters. The type of association is indicated by the *type* attribute, which may take such values as `exact` for exact equivalences, `uppercase` for uppercase equivalences, `lowercase` for lowercase equivalences, `standardized` for standardized forms, and `simplified` for simplified characters, etc., as in the following example:

```
<charDesc>
<char id="aenl">
<charName>LATIN LETTER ENLARGED SMALL A</charName>
<charProp>
<localName>entity</localName>
<value>aenl</value>
```

```
</charProp>
<mapping type="standardized">a</mapping>
</char>
</charDesc>
```

The mapping element may also be used to represent a mapping of the character or (more likely) glyph under discussion onto a character from the private use area as in this example:

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

A more precise documentation of the properties of any character or glyph may be supplied using the generic <charProp> element described in the next section. Despite its name, this element may be used for either characters or glyphs.

Character Properties The Unicode Standard documents 'ideal' characters, defined by reference to a number of properties (or attribute-value pairs) which they are said to possess. For example, a lower case letter is said to have the value Ll for the property general-category. The Standard distinguishes between normative properties (i.e. properties which form part of the definition of a given character), and informative or additional properties which are not normative. It also allows for the addition of new properties, and (in some circumstances) alteration of the values currently assigned to certain properties. When making such modifications however, great care should be taken not to over-ride standard informative properties for characters which exist in the Unicode Standard.See further the Unicode technical report TR23 Character Property Model

The <charProp> element allows an encoder to supply information about a character or glyph. Where the information concerned relates to a property which has already been identified in the Unicode Standard, encoders are urged to use the appropriate Unicode property name.

The following elements are used to record character properties:

- **<unicodeName>** contains the name of a registered Unicode normative or informative property.

  version specifies the version number of the Unicode Standard in which this property name is defined.

- **<localName>** contains a locally defined name for some property.
  - No attributes other than those globally available (see definition for tei.global.attributes)
- **<value>** contains a single value for some property, attribute, or other analysis.
  - No attributes other than those globally available (see definition for tei.global.attributes)

For each property, the encoder must supply either a <unicodeName> or a <localName>, followed by a <value>.

For convenience, we list here some of the normative character properties and their values. For full information, refer to chapter 4 of The Unicode Standard, or the online documentation of the Unicode Character Database.

general-category The general category (described in the Unicode Standard chapter 4 section 5) is an assignment to some major classes and subclasses of characters. Suggested values for this property are listed here:

Ll Letter, lowercase

Lu Letter, uppercase

Lt Letter, titlecase

Lm Letter, modifier

Lo Letter, other

Mn Mark, nonspacing

Mc Mark, spacing combining

Me Mark, enclosing

Nd Number, decimal digit

Nl Number, letter

No Number, other

Pc Punctuation, connector

Pd Punctuation, dash

Ps Punctuation, open

Pe Punctuation, close

Pi Punctuation, initial quote

Pf Punctuation, final quote

Po Punctuation, other

Sm Symbol, math

Sc Symbol, currency

Sk Symbol, modifier

So Symbol, other

Zs Separator, space

Zl Separator, line

Zp Separator, paragraph

Cc Other, control

Cf Other, format

Cs Other, surrogate

Co Other, private use

Cn Other, not assigned

directional-category This property applies to all Unicode characters. It governs the application of the algorithm for bi-directional behaviour, as further specified in Unicode Annex 9, The Bidirectional Algorithm. The following 19 different values are currently defined for this property in the Unicode Character Database:

AL right to left Arabic

AN Arabic Number

B Paragraph separator

BN Boundary Neutral

CS Common Number Separator

EN European Number

ES European Number Separator

ET European Number Terminator

L left to right

LRE left to right embedding

LRO left to right override

NSM Non-spacing Mark

ON Other neutrals

PDF Pop Directional Format

R right to left

RLE right to left embedding

RLO right to left override

S Segment separator

WS Whitespace

canonical-combining-class This property exists for characters that are not used independently, but in combination with other characters, for example the strokes making up CJK characters. It records a class for these characters, which is used to determine how they interact typographically. The following values are defined in the Unicode Standard 4.2:

0 Spacing, split, enclosing, reordrant, and Tibetan subjoined

1 Overlays and interior

7 Nuktas

8 Hiragana/Katakana voicing marks

9 Viramas

10 Start of fixed position classes

199 End of fixed position classes

200 Below left attached

202 Below attached

204 Below right attached

208 Left attached (reordrant around single base character)

210 Right attached

212 Above left attached

214 Above attached

216 Above right attached

218 Below left

220 Below

222 Below right

224 Left (reordrant around single base character)

226 Right

228 Above left

230 Above

232 Above right

233 Double below

234 Double above

240 Below (iota subscript)

character-decomposition-mapping This property is defined for characters, which may be decomposed, for example to a canonical form plus a typographic variation of some kind. For such characters the Unicode standard specifies both a decomposition type and a decomposition mapping (i.e. another Unicode character to which this one may be mapped in the way specified by the decomposition type). The following types of mapping are defined in the Unicode Standard:

font A font variant (e.g. a blackletter form)

noBreak A no-break version of a space or hyphen

initial An initial presentation form (Arabic)

medial A medial presentation form (Arabic)

final A final presentation form (Arabic)

isolated An isolated presentation form (Arabic)

circle An encircled form

super A superscript form

sub A subscript form

vertical A vertical layout presentation form

wide A wide (or zenkaku) compatibility character

narrow A narrow (or hankaku) compatibility character

small A small variant form (CNS compatibility)

square A CJK squared font variant

fraction A vulgar fraction form

compat Otherwise unspecified compatibility character

numeric-value This property applies for any character which expresses any kind of numeric value. Its value is the intended value in decimal notation.

mirrored The mirrored character property is used to properly render characters such as U+0028, OPENING PARENTHESIS independent of the text direction: it has the value Y (character is mirrored) or N (code is not mirrored).

the Unicode Standard also defines a set of informative (but non-normative) properties for Unicode characters. If encoders want to provide such properties, they may be included using the suggested Unicode name, tagged using the <unicodeName> element. However, encoders may also supply other locally-defined properties, which must be named using the <localName> element to distinguish them. If a Unicode name exists for a given property, it should however always be preferred to a locally defined name. Locally defined names should be used only for properties which are not specified by the Unicode Standard.

1.1.3 Annotating characters

Annotation of a character becomes necessary when it is desired to distinguish it on the basis of certain aspects (say, its graphical appearance) only. In a manuscript, for example, where distinctly different forms of the letter "r" can be recognized, it might be useful to distinguish them for analytic purposes, quite distinct from the need to provide a accurate representation of the page. A digital facsimile, particularly one linked to a transcribed and encoded version of the text, will always provide a superior visual representation, but cannot be used to support arguments based on the distribution of such different forms. Character annotation as described here provides a solution to this problem. It should be kept in mind that any kind of text encoding is an abstraction and an interpretation of the text at hand, which will not necessarily be useful in reproducing an exact facsimile of the appearance of a manuscript.

We begin by defining two distinct <glyph> elements, one for each of the forms of the letter we wish to distinguish:

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

With these definitions in place, occurrences of these two special "r"s in the text can be annotated using the element <g>:

```
<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>
```

As can be seen in this example, the <glyph> element pointed to from the <g> element will be interpreted as an annotation on the content of the element <g>. This mechanism can also be used to indicate ligatures, as in the following example:

```
<p> ...  <g ref="#Filig">Fi</g>lthy riches...</p>
<!- in the chardesc ->
<glyph id="Filig">
```

```
<glyphName>LATIN UPPER F AND LATIN LOWER I LIGATURE</glyphName>
<figure url="Filig.png"/>
</glyph>
```

(In fact the Unicode Standard does provide a character to represent the Fi ligature; the encoder may however prefer not to use it in order to simplify other text processing operations, such as indexing).

With this markup in place, it will be possible to write programs to analyze the distribution of the different letters "r" as well as produce more 'faithful' renderings of the original. It will also be possible to produce normalized versions by simple ignoring the annotation pointed to by the element <g>.

For brevity of encoding, it may be preferred to pre-define internal entities such as the following:

```
<!ENTITY r1 '<g ref="#r1">r</g>' >
<!ENTITY r2 '<g ref="#r2">r</g>' >
```

which would enable the same material to be encoded as follows:

```
<p>Wo&r1;ds in this manusc&r2;ipt are
sometimes written in a funny way.</p>
```

The same technique may be used to represent particular abbreviation marks as well as to represent other characters or glyphs. For example, if we believe that the r-with-one-funny-stroke is being used as an abbreviation for receipt, this might be represented as follows:

```
<abbr expan="receipt">&r1;</abbr>
```

Note however that this technique employs markup objects to provide a link between a character in the document and some annotation on that character. It cannot therefore be used in places where such markup constructs are not allowed, notably in attribute values. In the current P5 version of the TEI Guidelines, a systematic review of attribute datatypes has been carried out, and a large number of changes made. In cases where an attribute value might have occasion to include running text, and hence might need to use the mechanism discussed in this chapter, an alternative solution using child elements has been proposed.

1.1.4 Adding new characters

The creation of additional characters for use in text encoding is similar to the annotation of an existing character. The same element <g> is used to provide a link from the character instance in the text to the character definition in <charDesc> element. The main difference is that the <g> element now points to a <char> element. The element <g> itself will generally be empty, but could contain a codepoint from the Private Use Area (PUA) of the Unicode Standard, which is an area set aside for the very purpose of privately adding new characters to a document. Recommendations on how to use such PUA characters are given in the following section.

In some circumstances, it may be desirable to provide a single precomposed form of a character that is encoded in Unicode only as a sequence of codepoints. For example, in Medieval Nordic material, a character looking like a lower case letter Y with a dot and an acute-accent above it may be encountered so frequently that the encoder wishes to treat it as a distinct character. In the transcription concerned, the encoder enters this letter as &ydotacute;, which when the transcription is processed can then be expanded in one of two ways, depending on the DTD in force. It may be translated directly into a locally-defined PUA character (say &#xE0A4) for local processing only; alternatively it may be translated into the more portable representation

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

. In either case, the following new character definition should be supplied:

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

This definition specifies the mapping between this composed character and the individual Unicode-defined characters which make it up. It also supplies a single locally-defined property ('entity')for the character concerned, the purpose of which is to supply a recommended character entity name for the character.

Under certain circumstances, Han characters can be written within a circle. Rather than considering this as simply an aspect of the rendering, an encoder may wish to treat such circled characters as entirely distinct derived characters. For a given character (say that represented by the numeric-character reference 人) the circled variant might conveniently be represented as

```
<g ref="#U4EBA-circled"/>
```

, which references a definition such as the following:

```
<char id="U4EBA-circled">
<charName>CIRCLED IDEOGRAPH</charName>
<charProp>
<unicodeName>character-decomposition-mapping</unicodeName>
<value>circle</value>
</charProp>
<charProp>
<localName>daikanwa</localName>
<value>36</value>
</charProp>
<mapping type="standard">
&U+4EBA;
</mapping>
<mapping type="PUA">
&U+E000;
</mapping>
</char>
```

In this example, the 'circled ideograph' character has been defined with two mappings, and with two properties. The two properties are the Unicode-defined character-decomposition which specifies that this is a circled character, using the appropriate terminology (see 1.1.2.1. Character Properties above) and a locally defined property known as 'daikanwa'. The two mappings indicate firstly that the standard form of this character is the character 人, and secondly that the character used to represent this character locally is the PUA character . For convenience of local processing this PUA character may in fact appear as content of the <g> element. In general, however, the <g> element will be empty.

1.1.5 How to use codepoints from the 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 4.0), there are around 137,000 codepoints available in this area, which should be enough for most needs. No codepoint 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 codepoints defined by the Unicode Standard, PUA codepoints should *not* be used directly in documents intended for blind interchange. Instead of using PUA codepoints directly in the document content, entity references should be used. This will make it easier for receiving parties to find out what PUA characters are used in a document and where possible codepoint clashes with local use on the receiving side occurs.

In the two previous examples, we mentioned that the variant characters concerned might well be assigned specific codepoints from the PUA. This might, for example, facilitate the use of a particular font which displays the desired character at this codepoint in the local processing environment. Since however this assignment would be valid only on the local site, documents containing such codepoints are unsuitable for blind interchange. During the process of preparing such documents for interchange, any PUA codepoints must be changed either to an encoding such as <c ref="#xxxx"> or to an equivalent character entity reference, thus associating the character with an explicitly defined <char> element. The PUA character used during the preparation of the document might be recorded in the <char> element, as shown in the example in 1.1.4. Adding new characters; however there is no requirement that the same PUA character be used to represent it at the receiving site. Indeed, it may well be the case that this other site has already made an assignment of some other character to the original PUA codepoint.

For that reason, it is to be expected that a further translation into the local processing environment at the receiving site will be necessary to handle such characters, during which variant letters can be converted to hitherto unused codepoints on the basis of the information provided in the <charDesc> element.

This mechanism is rather weak in cases where DOM trees or parsed XML fragments are exchanged, which may increasingly be the case. The best an application can do here is to treat any occurrence of a PUA character only in the context of the local document and use the properties provided through the <char> element as a handle to the character in other contexts.

In the fullness of time, a character may become standardized, and thus assigned a specific code point outside the PUA. Documents which have been encoded using the mechanism must at the least ensure that this changed code point is recorded within the relevant <char> element; it will however normally be simpler to remove the <char> element and replace all occurrences of <g> elements which reference it by occurrences of the specific coded character.

1.1.6 Additional module defined by this section

The elements described in this section are declared in a module called gaiji which has the following formal declaration:

Module gaiji

```
datatype.ucs = token  pattern = "U\+([1-9A-F][0-9A-F]?)?[0-9A-F]4"

namespace local = ""
g = element g  g.content
g.content =
text,
tei.global.attributes,
g.attributes.ref,
g.newattributes,
[ a:defaultValue = "g" ] attribute TEIform  text ?
g.newattributes |= empty
tei.typed |= g
tei.seg |= g
g.attributes.ref = attribute ref  g.attributes.ref.content ?
g.attributes.ref.content = xsd:IDREF



char = element char  char.content
char.content =
( charName, gloss?, charProp*, desc?, mapping*, figure*, note?  ),
tei.global.attributes,
char.newattributes,
[ a:defaultValue = "char" ] attribute TEIform  text ?
char.newattributes |= empty

namespace local = ""
charName = element charName  charName.content
charName.content =
text,
tei.global.attributes,
charName.newattributes,
[ a:defaultValue = "charName" ] attribute TEIform  text ?
charName.newattributes |= empty
```

```
charProp = element charProp  charProp.content
charProp.content =
( ( unicodeName | localName ), value ),
tei.global.attributes,
charProp.newattributes,
[ a:defaultValue = "charProp" ] attribute TEIform  text ?
charProp.newattributes |= empty
tei.typed |= charProp

namespace local = ""
charDesc = element charDesc  charDesc.content
charDesc.content =
( desc?, ( char | glyph )+ ),
tei.global.attributes,
charDesc.newattributes,
[ a:defaultValue = "charDesc" ] attribute TEIform  text ?
charDesc.newattributes |= empty
tei.encoding |= charDesc


glyph = element glyph  glyph.content
glyph.content =
( name, gloss?, charProp*, desc?, mapping*, figure*, note?  ),
tei.global.attributes,
glyph.newattributes,
[ a:defaultValue = "glyph" ] attribute TEIform  text ?
glyph.newattributes |= empty

namespace local = ""
glyphName = element glyphName  glyphName.content
glyphName.content =
text,
tei.global.attributes,
glyphName.newattributes,
[ a:defaultValue = "glyphName" ] attribute TEIform  text ?
glyphName.newattributes |= empty


localName = element localName  localName.content
localName.content =
text,
tei.global.attributes,
localName.newattributes,
[ a:defaultValue = "localName" ] attribute TEIform  text ?
localName.newattributes |= empty
```

```
namespace local = ""
mapping = element mapping  mapping.content
mapping.content =
( ( text | c )* ),
tei.global.attributes,
mapping.newattributes,
[ a:defaultValue = "mapping" ] attribute TEIform  text ?
mapping.newattributes |= empty
tei.typed |= mapping



unicodeName = element unicodeName  unicodeName.content
unicodeName.content =
text,
tei.global.attributes,
unicodeName.attributes.version,
unicodeName.newattributes,
[ a:defaultValue = "unicodeName" ] attribute TEIform  text ?
unicodeName.newattributes |= empty
unicodeName.attributes.version =
attribute version  unicodeName.attributes.version.content ?
unicodeName.attributes.version.content = text


namespace local = ""
value = element value  value.content
value.content =
text,
tei.global.attributes,
value.newattributes,
[ a:defaultValue = "value" ] attribute TEIform  text ?
value.newattributes |= empty
```

1.2 Class catalogue

1.3 Pattern catalogue

1.3.1 datatype.ucs [macro]

Description: (Unicode Codepoint) Any legal Unicode codepoint, expressed using the Unicode notational convention of 'U+' followed by four to six hexadecimal digits, using the digits 0-9 and A-F (for 10 through 15, respectively). Use sufficient leading zeros to ensure there are at least four hexadecimal digits, but no more.

Declaration:

```
datatype.ucs = token  pattern = "U\+([1-9A-F][0-9A-F]?)?[0-9A-F]4"
```

Module: [gaiji]

1.4 Element catalogue

1.4.1 char [element]

Description: (character) provides descriptive information about a character which is not otherwise available in the document character set.

Declaration:

```
element char
tei.global.attributes,
( charName, gloss?, charProp*, desc?, mapping*, figure*, note?  )
```

Attributes: Global attributes only

Example:

```
<char id="U4EBA-circled">
<charName>CIRCLED IDEOGRAPH 4EBA</charName>
<charProp>
<unicodeName>character-decomposition-mapping</unicodeName>
<value>circle</value>
</charProp>
<charProp>
<localName>daikanwa</localName>
<value>36</value>
</charProp>
<mapping type="standard">
<g ref="U4EBA">[U+4EBA]</g>
</mapping>
</char>
```

Module: [gaiji]

1.4.2 charDesc [element]

Description: (character description) provides descriptive information about characters or glyphs additional to that implied by the current document character set.

Classes: tei.encoding

Declaration:

```
element charDesc  tei.global.attributes, ( desc?, ( char | glyph )+ )
```

Attributes: Global attributes and those inherited from tei.encoding

Example:

Module: [gaiji]

1.4.3 charName [element]

Description: (character name) contains the name of a character, expressed following Unicode conventions.

Declaration:

```
element charName  tei.global.attributes, text
```

Attributes: Global attributes only

Example:

```
<charName>CIRCLED IDEOGRAPH 4EBA</charName>
```

Notes:

The name must follow Unicode conventions for character naming. Projects working in similar fields are recommended to coordinate and publish their list of `<charName>`s to facilitate data exchange.

Module: [gaiji]

1.4.4 charProp [element]

Description: (character property) provides a name and value for some property of the parent character or glyph.

Classes: tei.typed

Declaration:

```
element charProp
tei.global.attributes,
( ( unicodeName | localName ), value )
```

Attributes: Global attributes and those inherited from tei.typed

Example:

```
<charProp>
<unicodeName>character-decomposition-mapping</unicodeName>
<value>circle</value>
</charProp>
<charProp>
<localName>daikanwa</localName>
<value>36</value>
</charProp>
```

Notes:

If the property is a Unicode Normative Property, then its `<unicodeName>` must be supplied. Otherwise, its name must be specied by means of a `<localname>`.

At a later release, additional constraints will be defined on possible value/name combinations using Schematron rules

Module: [gaiji]

1.4.5 g [element]

Description: (character or glyph) represents a non-standard character or glyph.

Classes: tei.typed tei.seg

Declaration:

```
element g  tei.global.attributes, g.attributes.ref, text
```

Attributes: (In addition to global attributes and those inherited from tei.typed, tei.seg)

ref points to a description of the character or glyph intended.

Datatype:
`xsd:IDREF`

Values: a pointer to some another element.

Example:

```
<c ref="#flig">fl</c>
```

Notes:

The name g is short for gaiji, which is the Japanese term for a non-standardized character or glyph.

Module: [gaiji]

1.4.6 glyph [element]

Description: (character glyph) provides descriptive information about a character glyph which is not otherwise available in the document character set.

Declaration:

```
element glyph
tei.global.attributes,
( name, gloss?, charProp*, desc?, mapping*, figure*, note?  )
```

Attributes: Global attributes only

Example:

```
<glyph id="r1">
<name>LATIN SMALL LETTER R WITH A FUNNY STROKE</name>
<charProp>
<localName>entity</localName>
<value>r1</value>
</charProp>
<figure url="glyph-r1.png"></figure>
</glyph>
```

Module: [gaiji]

1.4.7 glyphName [element]

Description: (character glyph name) contains the name of a glyph, expressed following Unicode conventions for character names.

Declaration:

```
element glyphName  tei.global.attributes, text
```

Attributes: Global attributes only

Example:

```
<glyphName>CIRCLED IDEOGRAPH 4EBA</glyphName>
```

Notes:

For characters of non-ideographic scripts, a name following the conventions for Unicode names should be chosen. For ideographic scripts, an Ideographic Description Sequence (IDS) as described in Chapter 10.1 of the Unicode Standard is recommended where possible. Projects working in similar fields are recommended to coordinate and publish their list of `<glyphName>`s to facilitate data exchange.

Module: [gaiji]

1.4.8 localName [element]

Description: (Locally-defined Property Name) contains a locally defined name for some property.

Declaration:

```
element localName  tei.global.attributes, text
```

Attributes: Global attributes only

Example:

```
<localName>daikanwa</localName>
<localName>entity</localName>
```

Notes:

No definitive list of local names is proposed. However, the name entity is recommended as a means of naming the property identifying the recommended character entity name for this character or glyph.

Module: [gaiji]

1.4.9 mapping [element]

Description: (character mapping) contains one or more characters which are related to the parent character or glyph in some respect, as specified by the *type* attribute.

Classes: tei.typed

Declaration:

```
element mapping  tei.global.attributes, ( ( text | c )* )
```

Attributes: Global attributes and those inherited from tei.typed

Example:

```
<mapping type="modern">r</mapping>
<mapping type="standard">
<g ref="U4EBA">[U+4EBA]</g>
</mapping>
```

Notes:

Suggested values for the *type* attribute include `exact` for exact equivalences, `uppercase` for uppercase equivalences, `lowercase` for lowercase equivalences, and `simplified` for simplified characters. The `<g>` elements contained by this element can point to either another `<char>` or `<glyph>`element or contain a character that is intended to be the target of this mapping.

Module: [gaiji]

1.4.10 unicodeName [element]

Description: (Unicode Property Name) contains the name of a registered Unicode normative or informative property.

Declaration:

```
element unicodeName
tei.global.attributes,
unicodeName.attributes.version,
text
```

Attributes: (In addition to global attributes)

version specifies the version number of the Unicode Standard in which this property name is defined.

Datatype:
`text`

Values: a valid version number.

Example:

```
<unicodeName>character-decomposition-mapping</unicodeName>
<unicodeName>general-category</unicodeName>
```

Notes:

A definitive list of current Unicode property names is provided in The Unicode Standard.

Module: [gaiji]

1.4.11 value [element]

Description: (value) contains a single value for some property, attribute, or other analysis.
Attributes: Global attributes only
Declaration:

```
element value  tei.global.attributes, text
```

Example:

```
<value>unknown</value>
```

Module: [gaiji]