The Association for Computers and the Humanities (ACH)

The Association for Computational Linguistics (ACL)

The Association for Literary and Linguistic Computing (ALLC)

Guidelines for Electronic Text Encoding and Interchange

Edited by

C. M. Sperberg-McQueen and Lou Burnard

TEI P2, Chapter 31 WD
Writing System Declaration
Text Encoding Initiative
Chicago, Oxford

September 20, 1993

Draft Version 2, September 20, 1993

Note

This is the current draft (as of September 20, 1993) of one section of the *Guidelines for Electronic Text Encoding and Interchange*, version 2 (also known as TEI P2), produced by the Text Encoding Initiative. This fascicle contains:

- a table of contents for TEI P2, indicating the status of each section as of the release date of this fascicle
- a draft for one section of TEI P2
- entries derived from this section for the reference list of tags of TEI P2 (if any)
- a User Response and Comment Form

Please feel free to send us your comments on the contents of this draft and their applicability to your needs. Comments may be sent to the TEI-L list server for general discussion, or directly to either of the TEI editors (adresses on the User Response and Comment form). Bear in mind in reading, however, that TEI P2 is a reference manual, not a tutorial in the use of electronic text, and unavoidably assumes either a certain amount of prior knowledge or patience with unfamiliar technical material. Note also that some matters which may appear obscure in one section may well be treated in more detail elsewhere in the full text of TEI P2.

TEI P2, like P1, is a working draft. The observant will find several typographic and presentational flaws in the current text: for example, cross-references are not properly resolved in the LaTeX versions and some parts of the reference material are not yet complete. We hope to rectify as many as possible of these flaws in the coming months, but have decided not to allow them to delay any further the release of the draft. We hope that those interested in the Guidelines will find the current state of the text readable and usable enough for the purpose of substantive comment. As far as possible, all comments received on this section will be taken into account in the preparation of the next version of the Guidelines, which is due by the end of 1993.

Updates on the status of the draft as a whole will be distributed automatically to all subscribers of the Listserv list TEI-L. To subscribe to TEI-L, send electronic mail to the address LISTSERV@UICVM (or Listserv@uicvm.uic.edu) containing the single line

```
subscribe tei-l J. Smith
```

(substituting your name for “J. Smith”). TEI-L is also the appropriate place to pose questions or offer public comments on the TEI guidelines and other relevant issues. The TEI-L file server contains all the sections of TEI P2 thus far released, as well as other TEI materials. Its contents are shadowed at various other sites around the world; for more information consult one of the following documents:

- TEI ED J8, “Obtaining the Second Version of the TEI Guidelines (TEI P2)” (describes how to retrieve electronic copies of TEI P2 and the various formats they are available in)
- TEI ED J9, “Obtaining Paper Copies of the Second Version of the TEI Guidelines (TEI P2)” (describes how to request paper copies of TEI P2, for those without electronic mail access)

These documents are available from the TEI-L file server or through the editors at the addresses given on the User Response and Comment Form.

Overall Table of Contents for TEI P2

The following list shows the expected contents of TEI P2 as a whole and indicates the current status of each chapter.

Contents

Part 5

Auxiliary Document Types

Chapter 27

Writing System Declaration

The *writing system declaration* or *WSD* is an auxiliary document which provides information on the methods used to transcribe portions of text in a particular language and script.[1] For purposes of the WSD, a *writing system* is a given method of representing a particular language, in a particular script or alphabet; the WSD specifies one method of representing a given writing system in electronic form. A single WSD thus links three distinct objects:

- the language in question
- the writing system (script, alphabet, syllabary) used to write the language
- the coded character set, entity names, or transliteration scheme used to represent the graphic characters of the writing system

Different natural languages thus have different writing system declarations, even if they use the same script. Different methods used to write the same language (e.g. Cyrillic or Latin encoding of Serbo-Croatian), and different methods of representing the same script in electronic form (e.g. different coded character sets such as ASCII or EBCDIC, or different transliteration schemes) similarly must use different writing system declarations.

This chapter describes first the overall structure of the WSD, and then documents the specific elements used to document the natural language, writing system, coded character sets, SGML entity names, and transliteration schemes united by the WSD. The concluding sections describe how the WSD is associated with different portions of a document. Standard pre-defined TEI writing system declarations, which should suffice for most uses, are also included. There follows a brief description of how to create a new WSD on the basis of an existing WSD. The final section provides a more formal discussion of the meaning of a WSD than is provided elsewhere.

27.1 Overall Structure of Writing System Declaration

A writing system declaration is a distinct auxiliary document, separate from any transcription for which it is used. It is encoded using a single **<writingSystemDeclaration>** element, and contains the following elements:

<writingSystemDeclaration> : declares the coded character set, transliteration scheme, or entity set used to transcribe a given writing system of a given language. Attributes include:
name : gives a formal name for the writing system declaration

[1] The primary responsibility for this chapter was borne by work group TR 1 (character set issues), chaired by Harry Gaylord. The original design of the writing system declaration and the first draft of its documentation were created by Steven J. DeRose; revisions have been made by Harry Gaylord and Syun Tutiya, among others.

date : gives the date on which the writing system declaration was last revised.

<language> : identifies the language being described in the writing system declaration.

<script> : contains a prose description of the script declared by a writing system declaration.

<direction> : specifies one or more conventional directions in which a language is written using a given script.

<characters> : contains a specification of the characters used in a particular writing system to write a particular language, and of how those characters are represented in electronic form.

<note> : (in a writing system) contains a note of any type.

All elements in the writing system declaration may bear either of the following two attributes:

id : gives a unique identifier for the element

lang : gives the language in which the content of the element is written.

These attributes function in the same way as the global *id* and *lang* attributes of the main TEI DTD (although for technical reasons the latter is declared differently): the former provides a unique SGML identifier for the element, and the latter identifies the language in which the contents of the element are expressed, using a code from ISO 639.

The overall structure of a writing system declaration is thus as follows:

```
<writingSystemDeclaration
          lang='eng'
          name=' ... '
          date='1993-05-29'>

<language iso639='...'>
    <!-- name of language here -->
    </language>
<script>
    <!-- description of script here ... -->
    </script>
<direction chars=LR lines=TB>
<characters>
    <!-- description of character inventory here ... -->
    </characters>
</writingSystemDeclaration>
```

The attributes *date* and *name* are required on the **<writingSystemDeclaration>** element. The *date* attribute is used to specify the date on which the WSD was written or last changed; this must be given in the format "yyyy-mm-dd", as defined by ISO 8601: 1988, *Data elements and interchange formats — Information interchange — Representation of dates and times*.

The *name* attribute is used to assign a formal name to the writing system declaration, for references to it from elsewhere. It is recommended, though not required, that the name be constructed as an SGML formal public identifier; for purposes of writing system declarations, this means it should follow the pattern of the following examples:

```
-//TEI P2: 1993//NOTATION WSD for Modern English//EN
-//WWP 1993//NOTATION WSD for 17th-century English//EN
-//OTA 1991//NOTATION WSD for Old English//EN
-//GLDV 1997//NOTATION WSD Mittelhochdeutsch//DE
```

That is, the identifier should consist of

1. either the string “-//” (indicating that the organization issuing the WSD is not registered with ISO) or the string “+//” (indicating that the organization is so registered)
2. a short string identifying the issuing organization and optionally the document which defines the WSD (or at least the issuing organization and the date)
3. the string “//”
4. the keyword “NOTATION”, indicating that in SGML terms, the writing system declaration functions as a non-SGML *notation*
5. a descriptive phrase indicating the contents of the WSD (which may, as here, begin “WSD for ...”)
6. the string “//”
7. the ISO 639 code for the language in which the WSD is written

The other elements of the WSD are described in the following sections.

The DTD for writing system declarations is included in file *teiwsd2.dtd*; a writing system declaration will thus begin with a document type declaration invoking that file:

```
<!DOCTYPE writingSystemDeclaration system 'teiwsd2.dtd' [
  <!-- entity sets are declared and invoked here -->

]>
```

The formal declaration of the writing system declaration is as follows:

```
<!-- 27.1:  Writing System Declaration                       -->
<!-- Text Encoding Initiative: Guidelines for Electronic      -->
<!-- Text Encoding and Interchange. DRAFT Version 2. 1992-93. -->

<!-- Copyright (c) 1990, 1992, 1993 ACH, ACL, ALLC.           -->
<!-- Permission to copy in any form is granted for use with   -->
<!-- TEI-aware systems and applications, provided this        -->
<!-- notice is included in all copies.                        -->

<!-- These materials may not be altered; modifications to     -->
<!-- these DTDs should be performed as specified in chapter   -->
<!-- MD of the Guidelines.                                    -->

<!-- These materials subject to revision. Current versions    -->
<!-- available from the Text Encoding Initiative.             -->
```

```
<!ENTITY % INHERITED '#IMPLIED' >
<!ENTITY % ISO-date 'CDATA' >
<!-- Embed entities for TEI generic identifiers.              -->

<!ENTITY % TEI.wsdNames system 'wdgis2.ent' >
%TEI.wsdNames;
<!ENTITY % a.global '
          id                 ID                  #IMPLIED
          lang               CDATA               %INHERITED'    >
<!ELEMENT writingSystemDeclaration
                        - -  (language, script, direction*,
                             characters, note*)                 >
<!ATTLIST writingSystemDeclaration
                             %a.global;
          name               CDATA               #REQUIRED
          date               %ISO-date           #REQUIRED      >
<!-- ... declarations from section 27.2                        -->
<!--     (Language identification)                             -->
<!--     go here ...                                           -->
<!-- ... declarations from section 27.3                        -->
<!--     (Script and writing direction)                        -->
<!--     go here ...                                           -->
<!-- ... declarations from section 27.4.1                      -->
<!--     (Base components)                                     -->
<!--     go here ...                                           -->
<!-- ... declarations from section 27.4.2                      -->
<!--     (Exceptions to the base components)                   -->
<!--     go here ...                                           -->
<!-- ... declarations from section 27.5                        -->
<!--     (Notes)                                               -->
<!--     go here ...                                           -->
```

27.2 Identifying the Language

The **<language>** element is used to name the language associated with the WSD. Its *iso639* attribute gives the ISO standard code for the language as defined by *ISO 639: 1988. Code for the representation of names of languages*.

<language> : identifies the language being described in the writing system declaration. Attributes include:
iso639 : gives the standard language code from ISO 639.

If the language in question is not included in the list in ISO 639, the value of the attribute *iso639* should be '' (the empty string).

The **<language>** element should not be confused with the global *lang* attribute; the element identifies the language whose writing system is being documented, while the attribute identifies the language in which the description is being written. A writing system declaration for classical Greek, for example, which itself is written in English, would have the value “eng” for the *lang* attribute on the top-level element, and the value “grc” for the *iso639* attribute on the **<language>** element:

```
<writingSystemDeclaration
          id='grc.beta'
          lang=eng
          name='-//TEI P2: 1993//NOTATION WSD for TLG Beta Code
               transliteration of ancient greek//EN'
          date='1993-05-29'>

<language iso639='grc'>Classical Greek.  This WSD documents the Beta
transcription code for classical Greek developed by the Thesaurus
Linguae Graecae of the University of California, Irvine.</language>
<!-- ... -->
</writingSystemDeclaration>
```

Normally, the language described is a natural language; in some cases, however, artificial languages, dialects, or other sublanguages may be usefully regarded as a language and documented in a writing system declaration. When a sublanguage is documented, a description of the sublanguage should be included in the **<language>** element:

```
 <language iso639='jpn'>
    Japanese (specialized writing system for waka)
 </language>
```

When a writing system declaration is prepared solely in order to document a coded character set or entity set suitable for use with many natural languages, the content of the **<language>** element should be “Various” (or the equivalent in the language of the WSD):

```
<writingSystemDeclaration
          lang=eng
          name='-//TEI P2: 1993//NOTATION WSD for ISO 646 IRV//EN'
          date='1993-05-29'>
  <language iso639=''>Various</language>
  <!-- ... -->
</writingSystemDeclaration>
```

The **<language>** element is formally defined thus:

```
<!-- 27.2:  Language identification                           -->
<!ELEMENT language      - O  (#PCDATA)                          >
<!ATTLIST language           %a.global;
          iso639             CDATA              #REQUIRED      >
<!-- This fragment is used in sec. 27.1                        -->
```

27.3 Describing the Writing System

The writing system itself is described in general terms using the following elements:

<script> : contains a prose description of the script declared by a writing system declaration.

<direction> : specifies one or more conventional directions in which a language is written using a given script. Attributes include:
- *chars* : indicates the order in which characters within a line are conventionally presented in this writing system. Suggested values include:
  - *LR* : left to right
  - *RL* : right to left
  - *TB* : top to bottom
  - *BT* : bottom to top
- *lines* : indicates the order in which lines conventionally follow each other in this writing system. Suggested values include:
  - *TB* : top to bottom
  - *BT* : bottom to top
  - *LR* : left to right
  - *RL* : right to left

The **<script>** element contains a prose description of the script, alphabet, syllabary, or other system of writing used to write the language in question. The **<direction>** element indicates the direction(s) in which the script is conventionally written. Both these elements are provided for the sake of human readers; neither is likely to be suited to machine processing without human intervention.

The Latin alphabet conventionally used to write English, for example, might be described thus:

```
<script>Latin alphabet (with diacritics for loan words)</script>
<direction chars=LR lines=TB>
```

The *chars* and *lines* attributes are used to indicate the direction in which characters within a line, and lines on the page, may legitimately be written using the script in question. If more than one direction is possible, the **<direction>** element may repeat or its attributes may be given more complex values. A script written vertically top to bottom, with lines arranged either left to right or right to left, for example, might be declared either thus:

```
<direction chars=TB lines=LR>
<direction chars=TB lines=RL>
```

or thus:

```
<direction chars=TB lines='LR RL'>
```

In very complex cases, the attributes may be given prose values:

```
<direction chars='boustrophedon:  LR, then RL, then LR, etc.'
           lines='TB'>
```

or the element may be omitted entirely (in which case experts on the script should be consulted for advice on proper processing):

```
<script>Japanese (mixture of kanji, hiragana, katakana)</script>
<!-- If you don't already know how Japanese is written,
     no DIRECTION element in the world will help you. -->
```

It should be noted that the **<direction>** element describes conventional display only: all scripts are subject to unusual treatment for aesthetic or other reasons, and such unusual treatment need not be foreseen here. (The Latin alphabet, for example, although conventionally written left-to-right, top-to-bottom, can be set vertically in signs or in other special cases.) Unusual methods of arranging the text on a page are best documented within the document instance by means of the global *rend* attribute.

The **<script>** and **<direction>** elements are declared thus:

```
<!-- 27.3:  Script and writing direction                        -->
<!ELEMENT script        - O  (#PCDATA)                            >
<!ATTLIST script             %a.global;                           >
<!ELEMENT direction     - O  EMPTY                                >
<!ATTLIST direction          %a.global;
          chars              CDATA              #REQUIRED
          lines              CDATA              #REQUIRED         >
<!-- This fragment is used in sec. 27.1                         -->
```

27.4 Documenting the Character Set and Its Encoding

27.4.1 Base Components of the WSD

The characters or graphic symbols of the writing system are documented in the **<characters>** element of the WSD. This documentation can take any of the following forms:

- reference to an international standard, national standard, or private coded character set
- reference to a public set of SGML entities
- reference to another WSD which documents the same script and the same methods of representing it electronically
- formal declaration of each graphic unit in the writing system
- a combination of the above: reference to one or more standard coded character sets, entity sets, or writing system declarations, followed by individual declaration of all exceptions

The coded character sets, entity sets, and external WSDs referred to are called the *base components* of the writing system declaration. The base components of a WSD are declared within the **<characters>** element using the following elements:

<characters> : contains a specification of the characters used in a particular writing system to write a particular language, and of how those characters are represented in electronic form.

<codedCharSet> : identifies a public or private coded character set which is used as a basic component of a writing system declaration.

<baseWsd> : identifies a writing system declaration whose mappings among characters, forms, entity names, and bit patterns are to be incorporated (possibly with modifications) in this writing system declaration.
<entitySet> : identifies a public or private entity set whose mappings between entity names and characters are to be incorporated (perhaps with modifications) into this writing system declaration.

The elements **<codedCharSet>**, **<baseWsd>**, and **<entitySet>** are all members of the class *baseStandard* and inherit from it the following attributes:

name : gives the normal citation form for the standard being referred to.
authority : indicates the authority responsible for issuing the standard being referred to: the TEI, the International Organization for Standardization (ISO), a national body, or a private body. Legal values are:
- *tei* : the base writing system declaration is a standard WSD issued by the Text Encoding Initiative
- *iso* : the character set or entity set was issued by ISO
- *national* : the character set or entity set was issued by a national standards body
- *private* : the writing system declaration, character set, or entity set was issued publicly by a private organization or project
- *none* : the writing system declaration, character set, or entity set has not been publicly issued by any organization; it is specific to an individual text or project

Some simple examples of the use of these elements follow:

```
<codedCharSet name='ANSI X3.4' authority='national'>
<codedCharSet name='ISO 646: 1991' authority='ISO'>
<entitySet    name='ISO 8879:1986//ENTITIES Added Latin 1//EN'
              authority='ISO'>
<baseWsd      name='-//TEI P2: 1993//WSD ISO 8859-1//EN'
              authority='TEI'>
```

The base components identify the set of characters used in the writing system, and further specify, for each character, the string(s) of bytes and entity names used to encode it in the text. This information may be modified by further information given within the **<exceptions>** element, as described below in section 27.4.2, “Exceptions in the WSD,” on page 11.

The elements for identifying the base components of the writing system declaration are declared thus:

```
<!-- 27.4.1:  Base components                                 -->
<!ELEMENT characters    - O  (codedCharSet*, baseWsd*,
                             entitySet*, exceptions?)         >
<!ATTLIST characters         %a.global;                       >
<!ENTITY % a.baseStandard '
          name               CDATA               #REQUIRED
          authority          (tei | iso | national | private |
                             none)               #REQUIRED'    >
<!ELEMENT codedCharSet  - O  EMPTY                            >
<!ATTLIST codedCharSet       %a.global;
```

```
                              %a.baseStandard;                  >
<!ELEMENT baseWsd       - O  EMPTY                              >
<!ATTLIST baseWsd            %a.global;
                             %a.baseStandard;                   >
<!ELEMENT entitySet     - O  EMPTY                              >
<!ATTLIST entitySet          %a.global;
                             %a.baseStandard;                   >
<!-- This fragment is used in sec. 27.1                        -->
```

27.4.2 Exceptions in the WSD

The **<exceptions>** element contains definitions for any character which differs in any respect from the specifications contained in the base components of the WSD. If no base components are named, then every character in the writing system must be defined explicitly.

The documentation for each character in the writing system indicates at least the following:

- the string of bytes used to represent the character
- whether the character is a letter, a punctuation mark, a diacritical mark, or falls into some other class
- a brief conventional name or description of the character
- any standard or local entity names used for the character
- the position of the character in the Universal Character Set (UCS) defined by ISO 10646, if known
- the position of the character's form in the tables prepared by the Association for Font Information Interchange

In addition, images of the character encoded in a graphics format or other notation may be associated with the character as internal or external figures. This information is encoded using the following elements:

<exceptions> : documents ways in which a writing system declaration differs from the coded character sets, base writing system declarations, and entity sets which form its bases.

<character> : defines one unit in a writing system, supplementing or overriding information provided in the base coded character sets, writing system declarations, and entity sets. Attributes include:

class : describes the function of the character using a prescribed classification. Legal values are:

lexical : character is used in writing words (lexical items) of the language (includes members of syllabaries and ideographic systems, as well as composite letter-plus-diacritic combinations)

punc : character is a punctuation mark which does not appear within lexical items

lexpunc : character can appear as a normal punctuation mark, but can also appear within a lexical item (and should usually, when occurring between two lexical characters, be treated as lexical—in English, hyphen and apostrophe are typically treated as members of this class)

digit : character is an Arabic decimal numeral (0, 1, ... 9) (does not include superscript numbers, circled numbers, numeric dingbats, etc.)
space : character represents some form of white space (space character, horizontal or vertical tab, newline, etc.)
dl : character is a diacritic applying to the following lexical character
ld : character is a diacritic applying to the preceding lexical character
dia : character is a diacritic which is explicitly joined to a lexical character by a joiner character
joiner : character is used to join a diacritic to the lexical character to which it applies (in some encoding schemes, the backspace control character may be used as a joiner; in others, a graphic character is used for the same function)
other : character does not fall into any of the other classes (dingbats and other unusual characters fall here)

<desc> : (in a writing system declaration) contains a description of a character or character form.

<form> : identifies one letter form taken by a particular character in a writing system declaration. Attributes include:
string : gives the byte string used to encode the letter form in the text.
codedCharSet : specifies which base coded character set the *string* value occurs in.
entityStd : gives the name of one or more entities defined for this character form in some standard entity set(s).
entityLoc : gives one or more entity names used locally for this character form.
ucs-4 : gives the position of the character form in the thirty-two bit “universal character set” defined by ISO 10646.
afiicode : gives one or more codes associated with this letter form by the Association for Font Information Interchange.

<figure> : (in a writing system declaration) contains an image of a character form, stored in-line in some declared notation. Attributes include:
notation : identifies the notation in which the figure is encoded.

<extFigure> : (in a writing system declaration) refers to a figure or illustration depicting the character form, which is stored in some declared notation external to the text. Attributes include:
notation : identifies the notation in which the figure is stored.
entity : gives the SGML name of the external entity which contains the figure.

The **<exceptions>** element contains a series of **<character>** elements only, each of which may contain descriptions of the character (including its name), notes, and a series of **<form>** elements documenting the different forms the character can take. Attributes on the **<character>** and **<form>** elements are used to convey the information mentioned above: byte string, entity names, UCS-4 code, etc.

A simple example:

```
<character class=lexical>
  <form string='A' ucs-4='0041' afiicode='0041'>
    <desc>Latin capital letter A</desc>
  </form>
</character>
```

When transliteration schemes are used, the *string* used to encode the character will typically be in a different alphabet:

```
<character class=lexical>
  <form string='*G' entityStd="Ggr" ucs-4='0393' afiicode='260044'>
    <desc>Greek capital letter Gamma</desc>
  </form>
</character>
```

The UCS-4 code is given as eight hexadecimal digits, one for each four bits of the thirty-two-bit value. For legibility a hyphen may be inserted as a separator after the fourth hexadecimal digit: “00000308” has the same meaning as “0000-0308”. Since in almost all cases at present the leading sixteen bits are zero, however, by convention the leading four hexadecimal zeros may be dropped entirely: the value “0308” is identical in meaning to the value “0000-0308”.

In some cases, the character is represented not as a single UCS character but as a sequence of such characters; in this case, each thirty-two-bit value except the last must be followed by a plus sign:

```
<character class='lexical'>
  <form string='*=+U'
        entityStd="Ucdgr"
        ucs-4='03A5+0302+0308'
        afiicode='F300EC'>
            <desc>Greek capital letter Upsilon with
                  circumflex and diaresis</desc>
  </form>
</character>
```

If a given **<character>** element has more than one encoding using ISO 10646 (e.g. both as “a-umlaut” and as “a” plus “umlaut”), then both encodings may be given, separated by blanks:

```
<character class='lexical'>
  <form string=''
        entityStd="Auml"
        afiicode='F127"
        ucs-4='0041+0308 00C4' >
            <desc>Latin capital letter A with umlaut</desc>
  </form>
</character>
```

In most cases, identifying the character or character form by means of its UCS-4 and AFII codes will suffice to identify the character for all later users of the WSD. In some cases, however, further information must be provided. This may be provided in a **<note>** attached to the **<character>** or **<form>** element:

```
<character class='lexical'>
  <form string='N'
        entityLoc="nn" >
        ucs-4="0274"   >
            <desc>Standard ms symbol for double n.</desc>
```

```
  </form>
  <note>This character has the form of a capital-letter N,
        but is written the same height as a lower-case N.
        Its appearance is thus that of UCS-4 0274, but it
        does not have the same semantics.
  </note>
</character>
```

In some cases, it will be necessary or useful to provide an image of the character in question, or to refer to a standard reference work for such an image. The following **<character>** element might be used to describe, for example, a common Old French abbreviation for “est”, for which the local entity *est* has been defined:

```
<character class='lexical'>
  <form string='' entityLoc="est">
     <desc>Old French abbreviation for 'est':  lowercase
           'e' with a tilde or macron above.</desc>
     <note>For an image of this character, see
           Cappelli, p. 113, column 1, line 4
           (leftmost and rightmost item).</note>
  </form>
</character>
```

Here, “Cappelli” is the name of a standard reference work which may be consulted to see what the character in question looks like.[2]

Where recourse to reference works is impossible, a picture of the character may be encoded using any standard graphics format, and associated with the character by standard SGML techniques. The SGML document must then have:

- an SGML notation declaration for the graphics format used
- an external entity declaration for the file containing the image
- an **<extFigure>** element to name the notation and the entity

For a discussion of graphic images and of the declaration of non-SGML notations, see chapter 24, “Tables, Equations, and Graphics,” [in separate fascicle]. If the Old French abbreviation is encoded using CGM (Computer Graphics Metafile) format in a file called “est.cgm”, then it may be associated with the appropriate character declaration as follows. In the DTD subset of the WSD, the following declarations are required:

```
<!NOTATION cgm PUBLIC 'ISO 8632/2//NOTATION
                       Computer Graphics Metafile
                       Character encoding//EN'>
<!ENTITY estFigure system 'est.cgm' NDATA cgm>
```

&@fn@sep

[2] *Dizionario di Abbreviature latine ed italiane* per cura di Adriano Cappelli, 6th ed. (Milan: Ulrico Hoepli, 1979). This work on Latin abbreviations might be less convenient for the purpose than one concentrating on Old French, but it is more widely used than any other.

In the body of the WSD itself:

```
<character class='lexical'>
  <form string=''
        entityLoc="est">
            <desc>Old French abbreviation for 'est':  lowercase
                  'e' with a tilde or macron above.
            </desc>
            <extFigure notation='cgm' entity='estFigure'>
            <note>For an image of this character, see
                  Cappelli, p. 110, column 1, line 4
                  (leftmost and rightmost item).
            </note>
  </form>
</character>
```

Despite now having a picture of the character, we retain the prose description and reference to Cappelli, for the sake of those without ready access to the appropriate graphics processors.

The **<exceptions>** element and its contents are declared thus:

```
<!-- 27.4.2:  Exceptions to the base components               -->
<!ELEMENT exceptions    - O  (character*)                       >
<!ATTLIST exceptions         %a.global;                         >
<!ELEMENT character     - O  (desc*, form+, note*)              >
<!ATTLIST character          %a.global;
          class              (lexical | punc | lexpunc | digit
                             | space | DL | LD | dia | joiner |
                             other)              lexical        >
<!ELEMENT desc          - O  (#PCDATA)                          >
<!ATTLIST desc               %a.global;                         >
<!ELEMENT form          - O  (desc+, (figure | extFigure)*,
                             note*)                             >
<!ATTLIST form               %a.global;
          string             CDATA               #IMPLIED
          codedCharSet       IDREF               #IMPLIED
          entityStd          ENTITIES            #IMPLIED
          entityLoc          ENTITIES            #IMPLIED
          ucs-4              CDATA               #IMPLIED
          afiicode           CDATA               #IMPLIED       >
<!ELEMENT figure        - -  CDATA                              >
<!ATTLIST figure             %a.global;
          notation           NAME                #REQUIRED      >
<!ELEMENT extFigure     - O  EMPTY                              >
<!ATTLIST extFigure          %a.global;
          notation           NAME                #REQUIRED
          entity             ENTITY              #REQUIRED      >
<!-- This fragment is used in sec. 27.1                         -->
```

27.4.3 Documenting Coded Character Sets and Entity Sets

Public or private coded character sets and entity sets may be usefully documented using WSDs; the WSD will make explicit some information (such as the UCS-4 and AFII codes) not normally given explicitly in character set standards or public entity sets. The coded character set or entity set being documented should be included by means of a **<codedCharSet>** or **<entitySet>** element; the **<exceptions>** element should include one **<character>** element for each character included in the character set or the entity set. Deciding whether to treat two entities or two bit patterns as separate characters or as forms of the same character will require knowledge of the script involved, and different encoders may reach different decisions. In cases of doubt, though, it is usually acceptable practice to treat each bit pattern in a coded character set, and each entity in an entity set, as a distinct character.

A non-standard local coded character set (e.g. an EBCDIC character set) may be documented in a WSD by defining one **<character>** element for each printable code point in the character set, adding the names of standard (and local) entities, UCS-4 codes, and AFII codes as appropriate. Since this extra information is useful in packing documents for interchange, and in processing *pattern* arguments in the TEI extended-pointer syntax described in section 16.2, “External Pointers and References,” [in separate fascicle], those responsible for a local installation are strongly encouraged to document the local system character set in a WSD, if it is not already so documented.

27.4.4 Documenting Transliteration Schemes

When a script is encoded not in a character set designed for it, but in one designed for another script, (e.g. Greek encoded using the Latin alphabet), a transliteration scheme is necessary. In documenting such a transliteration scheme, the coded character set actually in use should be named as a base component. An **<exceptions>** element can then be used to override the normal meaning of the individual byte strings used in the transliteration. For example, the following **<character>** element overrides the usual association of the byte representing “A” with the Latin letter “A” and substitutes instead an association with the Greek letter alpha:

```
<character class=lexical>
  <form string='A' entityStd="agr" ucs-4='03B1' afiicode='260061'>
    <desc>Greek small letter alpha</desc>
  </form>
</character>
```

Care should be taken in choosing or developing transliteration schemes to ensure that they are unambiguously reversible.

27.5 Notes in the WSD

Notes on the WSD, individual characters, or individual character forms may be included in the **<note>** element at the appropriate level.

<note> : (in a writing system) contains a note of any type.

Unlike its counterpart in the main TEI DTD, the **<note>** element within the writing system declaration may contain no paragraphs and no phrase-level elements: only character data. It is formally declared thus:

```
<!-- 27.5:  Notes                                                   -->
<!ELEMENT note          - O  (#PCDATA)                                >
<!ATTLIST note               %a.global;                               >
<!-- This fragment is used in sec. 27.1                             -->
```

27.6 Linkage between WSD and Main Document

The writing system declaration is associated with different portions of a main document by means of the global *lang* attribute. This attribute is defined as an SGML IDREF and its value must be the SGML identifier on a **<language>** element within the TEI header of the main document. The **<language>** element in turn provides, in its *wsd* attribute, the SGML name of the entity (usually an external file) containing the writing system declaration associated with that *lang* value.

At least one writing system declaration must be associated with any TEI document: this follows from the requirement that a value be specified for the *lang* attribute on the outermost element (**<tei.2>** or **<teiCorpus.2>**) of any TEI document, since the *lang* attribute is required to point at a **<language>** element in the TEI header, which in turn is required to indicate an entity containing the writing system declaration associated with that language.

27.7 Predefined TEI WSDs

The Text Encoding Initiative has defined standard writing system declarations for the following languages and scripts:

- The nine official languages of the European Community (Danish, Dutch, English, French, Greek, German, Italian, Portuguese, and Spanish), encoded using the character set ISO 8859-1
- Hebrew (using ISO 8859-8)
- Russian (using ISO 8859-5)
- classical Greek (using the Thesaurus Linguae Graecae's Beta Code transliteration scheme)

Work is underway on a standard writing system declarations for Japanese; resources permitting, declarations for Korean and Chinese will also be made.

In addition to the language-specific WSDs just mentioned, the TEI has defined a number of WSDs to document specific public coded character sets and entity sets.

These are used as components in the language-specific WSDs, and may be similarly used for locally developed WSDs. WSDs for specific ISO standard character sets include:

- ISO 646 (non-national subset)
- ISO 646 (International Reference Version)
- ISO 8859-1 (Latin 1, Western Europe)
- ISO 8859-2 (Latin 2, Eastern Europe)
- ISO 8859-5 (Latin and Cyrillic)
- ISO 8859-7 (Latin and Greek)
- ISO 8859-8 (Latin and Hebrew)
- ISO 8859-9 (Western Europe and Turkey)

Writing system declarations are also under preparation for a number of commercial character sets in wide use:

- IBM Code Page 437 (IBM PC, early models)
- IBM Code Page 850 (IBM PS/2 and later-model IBM PCs)
- IBM Code Page 1014
- Apple Macintosh default system character set
- Adobe Postscript default character set

The Text Encoding Initiative has prepared entity sets for use in transcribing some languages; these are distributed both as SGML entity sets and as TEI writing system declarations documenting those entity sets:

- -//TEI P2: 1993//ENTITIES Arabic//EN
- -//TEI P2: 1993//ENTITIES Coptic//EN
- -//TEI P2: 1993//ENTITIES Classical Greek//EN
- -//TEI P2: 1993//ENTITIES International Phonetic Alphabet//EN

WSDs will also be prepared for other standard entity sets, including:

- ISO 8879: 1986//ENTITIES Added Latin 1//EN
- ISO 8879: 1986//ENTITIES Added Latin 2//EN
- ISO 8879: 1986//ENTITIES Russian Cyrillic/EN
- ISO 8879: 1986//ENTITIES Non-Russian Cyrillic//EN
- ISO 8879: 1986//ENTITIES Greek Letters//EN
- ISO 8879: 1986//ENTITIES Diacritical Marks//EN
- ISO 8879: 1986//ENTITIES Box and Line Drawing//EN
- ISO 8879: 1986//ENTITIES Numeric and Special Graphic//EN
- ISO 8879: 1986//ENTITIES Publishing//EN
- ISO 8879: 1986//ENTITIES General Technical//EN

All TEI writing system declarations are distributed with the TEI document type definitions.

The standard TEI writing system declarations are expected to meet the needs of many encoders; some, however, will need to prepare new WSDs to describe character-encoding schemes not included in the standard WSDs.

27.8 Details of WSD Semantics

This section describes the meaning of the WSD in more formal terms than have been used elsewhere in this chapter; it can be skipped by most readers, but should be read carefully by those who wish to write complex writing system declarations or to implement software to process writing system declarations or to interpret them in the processing of TEI-conformant documents.

27.8.1 WSD Semantics: General Principles

A writing system declaration provides a complicated bundle of mappings:

- a 1:1 partial function from strings in given coded character sets to character forms
- a function from entity names to character forms, and therefore derivatively:
  - a function from entity names to strings
- a function from character forms to characters, and therefore derivatively:
  - a function from strings to characters
  - a function from entity names to characters
- a relation between UCS-4 codes and character forms
- a relation between AFII codes and character forms
- a function from UCS-4 codes to characters
- a relation from AFII codes to characters

To ensure that the relations described as functions are in fact functional, the following constraints apply on the WSD:

- No two **<form>** elements can have the same values for both *codedCharSet* and *string*. Since usually there is only one **<codedCharSet>** used as a basic component, this usually means each *string* attribute value must be unique in the WSD.
- No two **<form>** elements can name the same entity in either *entityStd* or *entityLoc*. (It is legal, though pointless, for both *entityStd* and *entityLoc* on the same **<form>** element to name the same entity.)
- More than one **<form>** element may have the same *UCS-4* value, but if so they must be within the same **<character>** element.

These constraints may be summarized thus: one "character" (however the creator of the WSD defines a character) can be associated with more than one byte string, entity name, UCS-4 code, or AFII code, but any single byte string (given a specific coded character set), any single entity name, and any single UCS-4 code must be associated with only one single **<character>**. One can, for example, associate both "tilde" and "logical not" with a **<character>** meaning "logical negation", but one cannot associate both a **<character>** called "tilde" and one called "logical negation" with the ASCII character 7/14: given a 7/14 in the text, it must be unambiguously clear whether the character is a "tilde" or a "logical negation". If one wishes to retain the ambiguity, one must define a **<character>** called (for example) "logical-not or tilde or swung-dash". Similar restrictions apply to entity names and UCS-4 codes: each must be associated with a single **<character>** element.

27.8.2 Semantics of WSD Base Components

The effects of naming coded character sets, entity sets, and other WSDs as base components may now be defined thus:

- reference to a coded character set makes available the set of bit-pattern-to-character mappings defined in the coded character set. I.e. if a WSD refers to a coded character set, then whenever the WSD is in use, any character in that coded character set may be used with its standard meaning (unless it has been redefined using the **<exceptions>** element). It is recommended that a WSD be provided for each coded character set, to make the mappings fully explicit.
- reference to an entity set makes available the set of entity-name-to-character mappings defined in the entity set. (It is assumed that standard public entity sets contain enough information to count as a valid mapping; for private entity sets, the preferred method of providing the necessary information is to define the entity set in a WSD). If for example a WSD refers to the ISO Latin 1 entity set, then whenever that WSD is in use, any entity in that set may be used with their public meaning, unless they have been redefined in the **<exceptions>** element.
- reference to a WSD makes available the set of mappings declared in that WSD; the language and writing system direction information given in the base WSD is ignored.

If reference is made only to standard character sets and entity sets, there is no mechanical method of associating the “characters” involved in one mapping with those involved in another. E.g. a reference to ISO 646 IRV provides a map from code point 5/11 to a character one might call “left square bracket”. A reference to entity set *ISO-pub1* provides a map from the entity name *lbr* to what should probably be considered the same character. There is however no guarantee that any processing software will necessarily be sufficiently intelligent to make this association of mappings automatically; it requires hard-coded knowledge of the specifics of certain character sets and entity sets.

When, however, base WSDs are used to document important entity sets and character sets, it does become possible to define mechanical methods of associating **<character>** elements in different base components.

27.8.3 Multiple Base Components

When multiple bases of the same type are referred to, the effects are these:

- if more than one coded character set is named, then it is expected that character-set shifting as described in ISO 2022 or some equivalent is in use, and proper shifting is the responsibility of the user. All strings in the WSD must specify the ID of the proper coded-character-set base, using the *codedCharSet* attribute.
- if more than one entity set is named, then entity names from all named sets may be used as values of the *entityStd* and *entityLoc* attributes. If the same name occurs in more than one entity set, the assumption is made that it refers each time to the same character.

- if more than one base WSD is named, then all characters declared in all the WSDs are available. For this case, we can define what happens to merge the different base components more precisely than for the other types of base component.
  — Any two **<form>** elements which name the same entity or the same string in the same coded character set are considered the same form, and are merged as described below in section 27.8.5, "Merger of Form and Character Elements," on page 24.
  — any two **<form>** elements which give the same UCS-4 code are considered forms of the same **<character>**, and their parent **<character>** elements are merged. The forms themselves may be merged or may remain distinct: if the forms have conflicting values for any attribute, they must remain distinct; if they don't conflict, they may be merged, at the option of the processing software. In the general case, there might be more than one way to perform mergers, so merger is not required.

The result of invoking multiple base WSDs is thus a merged WSD in which the **<form>** and **<character>** elements have been merged as prescribed. If the merger is impossible because the two WSDs are incompatible, a semantic error occurs. A set of WSDs is compatible and may be invoked together if all of the following are true:

- any given entity name is associated with a single string (in a given coded character set) and a single character class

- any given string or UCS-4 code is associated with a single character class

27.8.4 Semantics of Exceptions

We can now define the semantics of the **<exceptions>** element this way:

The base components provide a preliminary set of mappings, as described above. For convenience let us call this the *default map*. The **<exceptions>** element allows the user to modify the default map by defining further mappings and by overriding parts of the default map. There are three cases: a new **<character>** element replaces an old one, is merged with an old one, or is added to the set without affecting any old ones.

Case 1: If a **<form>** element within **<exceptions>** (F-new) "collides" with a **<form>** element in the default map (F-old), then the parent **<character>** element of F-new replaces the parent element of F-old. Two **<form>** elements collide if they have the same values for *codedCharSet* and *string*. (N.B. if this condition occurs within the default map, the two **<form>** elements are merged.)

E.g. to define the TLG Beta code transliteration of alpha as "a" we may first name ISO 646 IRV as a base component; this has the effect of creating the following (possibly imaginary) **<form>** element:

```
<character id=a class=lexical>
  <form string='a'>
    <desc>lowercase latin letter A</desc>
  </form>
</character>
```

We then include the following within the **<exception>** element:

```
<character id=alpha class=lexical>
  <form string='a' entityStd='gkalpha'>
    <desc>lowercase Greek alpha</desc>
  </form>
</character>
```

This overrides the **<character>** element for latin A, and indicates that in the transliteration scheme documented by this WSD, character 6/01 represents a Greek alpha, no matter what ISO 646 says.

Case 2: If a **<character>** element within **<exceptions>** “overlaps” with one in the default map, then the two **<character>** elements are merged. Two **<character>** elements overlap if any of their children name the same entity or UCS-4 code. (N.B. if these conditions occur within the default map, they lead to merger either of the two **<form>** elements — for entity name overlap — or of the two **<character>** elements.)

For example: suppose we wish to document the three-Rs transcription described in section 4.1.2, “Characters Not Available Locally,” [in separate fascicle]. We name ISO 646 IRV as a base character set (or WSD) and add the following exceptions:

```
<exceptions>
  <character id=r class=lexical>
      <desc>lowercase latin letter r</desc>
      <form string='' entityLoc='r' ucs-4='0072'>
          <desc>'normal' form, similar to modern print r and to
                Cappelli, p. 318, line 2, items 3, 6, 15.</desc>
      </form>
      <form string='' entityLoc='r2' ucs-4='0072'>
          <desc>'round' form, usually following 'o', similar
                to a modern Arabic digit 2 (or to Cappelli, p. 318,
                line 2, items 13 and 14)</desc>
      </form>
      <form string='' entityLoc='r2' ucs-4='0072'>
          <desc>'small-cap' form, like a capital R but
                same height as lowercase (cf. Cappelli, p. 318,
                line 1, items 2 and 3)</desc>
      </form>
  </character>
</exceptions>
```

As a second example, imagine we wish to document a local entity set for Old English in which we define local entities t (thorn), d (eth) and a (aesc). Assuming the TEI has provided a WSD for the Latin 1 entities, the whole WSD is this:

```
<writingSystemDeclaration
       name='-//OTA 1990//NOTATION WSD Old English entities//EN'
       date='1993-05-25'
       lang=eng>
  <language iso639=''>Various</language>
  <script>Latin alphabet, extended</script>
  <direction lines=TB chars=LR>
  <characters>
        <baseWsd name='-//TEI P2: 1993//NOTATION
                       WSD ISO Added Latin 1//EN'
                 authority=TEI>
        <exceptions>
            <character class=lexical>
                <form entityStd='thorn' entityLoc='t'>
                    <desc>lowercase latin letter thorn</desc>
                    </form></character>
            <character class=lexical>
                <form entityStd='Thorn' entityLoc='T'>
                    <desc>uppercase latin letter thorn</desc>
                    </form></character>
            <character class=lexical>
                <form entityStd='eth' entityLoc='d'>
                    <desc>lowercase latin letter eth</desc>
                    </form></character>
            <character class=lexical>
                <form entityStd='Eth' entityLoc='D'>
                    <desc>uppercase latin letter eth</desc>
                    </form></character>
            <character class=lexical>
                <form entityStd='aelig' entityLoc='a'>
                    <desc>lowercase latin aesc (= digraph aelig)</desc>
                    </form></character>
            <character class=lexical>
                <form entityStd='AElig' entityLoc='A'>
                    <desc>uppercase latin aesc (= digraph aelig)</desc>
                    </form></character>
        </exceptions>
  </characters>
  <note>This WSD is just to document the local entities; it should be
        named as a base WSD by the actual writing system declaration.
  </note>
</writingSystemDeclaration>
```

This has the effect of merging the **<character>** elements for *thorn*, *eth*, and *aesc* (or a-e ligature) defined in the ISO Latin 1 WSD with those given here, which specify the local entity name. The **<form>** elements may or may not be merged, so the software may or may not actually realize that the local entity *t* corresponds with the UCS-4 code given in the TEI WSD for ISO Latin 1.

The full local WSD can then be this:

```
<writingSystemDeclaration
       name='-//OTA 1993//NOTATION Old English WSD//EN'
       date='1993-05-25'
       lang=eng>
  <language iso639=ang>Anglo-Saxon / Old English</language>
  <script>Latin alphabet, extended</script>
  <direction lines=TB chars=LR>
  <characters>
        <baseWsd name='-//TEI P2: 1993//NOTATION WSD ISO 646 IRV//EN'
                 authority='TEI'>
        <baseWsd name='-//OTA 1990//NOTATION
                       WSD Old English entities//EN'
                 authority='private'>
        <baseWsd name='-//TEI P2: 1993//NOTATION
                       WSD ISO Added Latin 1//EN'
                 authority='TEI'>
  </characters>
</writingSystemDeclaration>
```

We refer explicitly to ISO Latin 1, for clarity, but in theory it has already been included in "-//OTA 1990//WSD Old English entities//EN" and need not be repeated. At this time, the rules for merger would force our local **<form>** elements to be merged with the standard **<form>** elements, so the local entity *t* would map correctly into the UCS-4 character set.

Case 3: If a **<character>** element has no **<form>** children which collide with anything in the default map, and does not itself overlap with anything in the default map, then it is simply added to the default map.

For example, suppose we wish to document an abbreviation used for Old French "est" in our manuscript, which resembles e with a tilde or macron. Since we expect we may have more abbreviations for "est", we use the local entity name "est1" for this one. Within **<exceptions>**, we declare the abbreviation thus:

```
<character id=est1 class=lexical>
  <form string='' entityLoc='est1'>
    <desc>abbreviation for 'est', lowercase latin e
          with a tilde or macron above, similar to
          Cappelli p. 113, col 1, items 4(a) and 8.</desc>
  </form>
</character>
```

27.8.5 Merger of Form and Character Elements

In some cases, the **<form>** and **<character>** elements introduced notionally by reference to a coded character set or entity set, or introduced explicitly by reference to a base WSD, may be considered as referring to identical objects; this is called *merger*. Two **<form>** elements F1 and F2 can be merged if they both have the same values for *codedCharSet* and *string*, or if *codedCharSet* and *string* are unspecified (implied) in at least one. When F1 and F2 are merged, the result is a (possibly imaginary) **<form>** element (F3) the attributes of which are derived thus:

- if F1 has no value for a *codedCharSet*, then F3 has the same value for this attribute as does F2. (If both F1 and F2 have explicit values, the values must be identical.)
- if F1 has an empty string for *string*, then F3 has the same value as F2. (If both have values other than '', they must be identical.)
- for *entityStd*, *entityLoc*, *ucs-4*, and *afiiCode*, F3 gets a value containing all the entity names or codes which appear in the corresponding attribute values of either F1 or F2. (I.e. the attribute values are viewed as sets, and F3 gets the union of F1 and F2.)

The children of the new element are derived by taking all the **<desc>** children of F1, then all the **<desc>** children of F2; all the **<figure>** children of F1, then those of F2; all the **<note>** children of F1, then all the **<note>** children of F2. In other words, all the children of the source elements survive as children of the result element.

Two **<character>** elements C1 and C2 may be merged unless their values for *class* differ. The resulting **<character>** element C3 has the same **<class>** value as C1 and C2, and all the children of C1 and C2 are made children of C3 (**<desc>** children first, then **<form>** children).

Note that merger is sometimes required by the semantic rules given above, and sometimes optional. If merger is required but not legal (because the two elements to be merged are incompatible), then a semantic error has occurred and the two base WSDs which give rise to it should not be invoked together.

Part 7

Alphabetic Reference List of Tags

Class Name: **baseStandard** (i.e. base-component standard)
Description: groups elements in a writing system which refer to some public or private standard as part of the basis for the writing system declaration
Attributes:

name: gives the normal citation form for the standard being referred to.
Data type: CDATA
Value: For national and international standards, the value should be the normal citation form for the standard; for public entity sets, it should be the standard public entity text; for private character sets, WSDs, and entity sets, it is recommended that the form of SGML *public identifiers* be used.
Default value: #REQUIRED
Example:

```
<codedCharSet name='ANSI X3.4' authority='national'>
<codedCharSet name='ISO 646: 1991' authority='ISO'>
<entitySet    name='ISO 8879:1986//ENTITIES Added Latin 1//EN'
              authority='ISO'>
<baseWsd      name='-//TEI P2: 1993//WSD ISO 8859-1//EN'
              authority='TEI'>
```

Remarks:
This attribute is required.

authority: indicates the authority responsible for issuing the standard being referred to: the TEI, the International Organization for Standardization (ISO), a national body, or a private body.
Data type: (tei | iso | national | private | none)
Legal values are:
tei : the base writing system declaration is a standard WSD issued by the Text Encoding Initiative
iso : the character set or entity set was issued by ISO
national : the character set or entity set was issued by a national standards body
private : the writing system declaration, character set, or entity set was issued publicly by a private organization or project
none : the writing system declaration, character set, or entity set has not been publicly issued by any organization; it is specific to an individual text or project
Default value: #REQUIRED
Example:

Remarks:
This attribute is required.

Remarks:
Part: auxiliary tag set for writing system declarations.
Member of classes:
DTD file:
Discussed in 27.4.1, “Base Components of the WSD,” on page 9.

<baseWsd> (i.e. base writing system declaration)

Description: identifies a writing system declaration whose mappings among characters, forms, entity names, and bit patterns are to be incorporated (possibly with modifications) in this writing system declaration.
Attributes: [None: global and inherited attributes only.]
Example:

Remarks: Reference in a WSD to a base WSD makes available the set of mappings among characters, forms, entity names, and bit patterns declared in the base WSD; the language and writing system (direction) information given in the base WSD is ignored.
Part: auxiliary tag set for writing system declarations.
Member of classes: basestd
DTD file:
Data description: Empty.
May occur within: characters
May contain: EMPTY
Declarations:

```
<!ELEMENT baseWsd       - O  EMPTY                         >
<!ATTLIST baseWsd            %a.global;
                             %a.baseStandard;              >
```

Discussed in 27.4.1, “Base Components of the WSD,” on page 9.
This tag is optional.

<character>

Description: defines one unit in a writing system, supplementing or overriding information provided in the base coded character sets, writing system declarations, and entity sets.
Attributes:

class: describes the function of the character using a prescribed classification.

Data type: (lexical | punc | lexpunc | digit | space | DL | LD | dia | joiner | other)

Legal values are:

lexical : character is used in writing words (lexical items) of the language (includes members of syllabaries and ideographic systems, as well as composite letter-plus-diacritic combinations)

punc : character is a punctuation mark which does not appear within lexical items

lexpunc : character can appear as a normal punctuation mark, but can also appear within a lexical item (and should usually, when occurring between two lexical characters, be treated as lexical—in English, hyphen and apostrophe are typically treated as members of this class)

digit : character is an Arabic decimal numeral (0, 1, ... 9) (does not include superscript numbers, circled numbers, numeric dingbats, etc.)

space : character represents some form of white space (space character, horizontal or vertical tab, newline, etc.)

dl : character is a diacritic applying to the following lexical character

ld : character is a diacritic applying to the preceding lexical character

dia : character is a diacritic which is explicitly joined to a lexical character by a joiner character

joiner : character is used to join a diacritic to the lexical character to which it applies (in some encoding schemes, the backspace control character may be used as a joiner; in others, a graphic character is used for the same function)

other : character does not fall into any of the other classes (dingbats and other unusual characters fall here)

Default value: lexical

Example:

Remarks: The classification of characters provided by this attribute serves both informative and normative purposes: it helps identify the character being described, and the classification is used to define the meaning of the special character-class codes in the TEI extended pointer syntax described in chapter 16, “Segmentation and Alignment,” [in separate fascicle].

This attribute is required.

Example:

Remarks: The notion of “characters” as units in a writing system is widely spread, but not consistently defined; the **<character>** element should be used to identify whatever units the encoder wishes to distinguish as the meaningfully distinct graphic units of the writing system. In most cases, these will correspond to the units of coded character sets, but that this is not a requirement: a-umlaut, for example, may be treated as one character or two, depending on the user's preference, regardless of how the coded character set in use treats it. In most cases, also, the units distinguished by the **<character>** element will be the “graphemic” units of the writing system in question; however, since experts disagree on whether items like umlaut (let alone a given set of Chinese characters with regional variations in China, Korea, and Japan) are best treated as distinct graphemes or not, the association of **<character>** elements with the graphemes of a writing system provides at most a heuristic device for making reasonable decisions, rather than a definitive unambiguous test.

Different forms of the same “character” may be distinguished for whatever reason, as in the three-R example of chapter 4, “Characters and Character Sets,” [in separate fascicle]. In this case the different letter forms are distinguished by documenting them in different **<form>** elements; the fact that the different letter shapes do not make a lexical difference in the text may be expressed by grouping all three letter forms under the same **<character>** element. (Alternatively, the three forms may be treated as three distinct characters, for convenience or for whatever reason, by defining a distinct **<character>** element for each.)

Part: auxiliary tag set for writing system declarations.
Member of classes:
DTD file:
Data description: May contain one or more description elements (optional), a series of one or more **<form>** elements identifying different forms of the character, and an optional series of notes.
May occur within: exceptions
May contain: desc form note
Declarations:

```
<!ELEMENT character     - O  (desc*, form+, note*)            >
<!ATTLIST character          %a.global;
          class              (lexical | punc | lexpunc | digit
                             | space | DL | LD | dia | joiner |
                             other)              lexical       >
```

Discussed in 27.4.2, “Exceptions in the WSD,” on page 11.
This tag is optional.

<characters>

Description: contains a specification of the characters used in a particular writing system to write a particular language, and of how those characters are represented in electronic form.
Attributes: [None: global and inherited attributes only.]
Example:

```
<writingSystemDeclaration
    name='-//TEI P2: 1993//WSD ISO 8859-1//en'
    date='1993-06-01'>
    <language iso639=''>various</>
    <direction chars=LR lines=TB>
    <characters>
         <codedCharSet name='ISO 8859-1: 1992' authority='ISO'>
         <exceptions>
              <!-- ... -->
         </exceptions>
    </characters>
</writingSystemDeclaration>
```

Remarks:
Part: auxiliary tag set for writing system declarations.
Member of classes:
DTD file:
Data description: May contain a series of base components (coded character sets, writing system declarations, and entity sets) followed by a set of exceptions to define differences from the base components.
May occur within: writingSystemDeclaration
May contain: baseWsd codedCharSet entitySet exceptions
Declarations:

```
<!ELEMENT characters    - O  (codedCharSet*, baseWsd*,
                              entitySet*, exceptions?)            >
<!ATTLIST characters         %a.global;                           >
```

Discussed in 27.4.1, “Base Components of the WSD,” on page 9.
This tag is required.

<codedCharSet> (i.e. base coded character set)

Description: identifies a public or private coded character set which is used as a basic component of a writing system declaration.
Attributes: [None: global and inherited attributes only.]
Example:

Remarks: Reference to a coded character set makes the set of bit-pattern-to-character mappings defined in the coded character set available for use in text encoded with the writing system declaration. Unless the character inventory and mappings are modified by the **<exceptions>** element, any character in any base coded character set may be used with its standard meaning within text to which the writing system declaration applies.

In order to make the mappings of bit patterns to characters more explicit than they sometimes are in character-set standards, the TEI will provide standard writing system declarations which document the bit-pattern-to-character mappings defined by various commonly used coded character sets.

Part: auxiliary tag set for writing system declarations.
Member of classes: basestd
DTD file:
Data description: An empty element.
May occur within: characters
May contain: EMPTY
Declarations:

```
<!ELEMENT codedCharSet  - O  EMPTY                          >
<!ATTLIST codedCharSet       %a.global;
                             %a.baseStandard;               >
```

Discussed in 27.4.1, “Base Components of the WSD,” on page 9.
This tag is optional.

<desc> (i.e. description)

Description: (in a writing system declaration) contains a description of a character or character form.
Attributes: [None: global and inherited attributes only.]
Example:

```
<form string='a'>
  <desc>Latin lower-case letter A</desc>
<form>
<form string='' entityLoc='infinity'>
  <desc>author's private symbol for infinity:  a parabola open to
        the right, with a dot at its focal point.  In general, the
        character rests on the base line and rises to about the
        height of a typical ascender.</desc>
</form>
```

Remarks: The **<desc>** element should usually contain the name of the character or character form; in some cases further information (e.g. the description of the character's shape) will be useful, as shown in the example.
Part: auxiliary tag set for writing system declarations.
Member of classes:
DTD file:
Data description: May contain character data only.
May occur within: character form
May contain: #PCDATA
Declarations:

```
<!ELEMENT desc          - O  (#PCDATA)                      >
<!ATTLIST desc               %a.global;                     >
```

Discussed in 27.4.2, “Exceptions in the WSD,” on page 11.
This tag is optional.

<direction>

Description: specifies one or more conventional directions in which a language is written using a given script.

Attributes:

chars (i.e. characters): indicates the order in which characters within a line are conventionally presented in this writing system.

Data type: CDATA

Suggested values include:

LR : left to right
RL : right to left
TB : top to bottom
BT : bottom to top

Default value: #REQUIRED

Remarks: If the characters follow each other in some order other than top-to-bottom, bottom-to-top, left-to-right, or right-to-left, then some value other than those indicated should be used.

This attribute is required.

lines: indicates the order in which lines conventionally follow each other in this writing system.

Data type: CDATA

Suggested values include:

TB : top to bottom
BT : bottom to top
LR : left to right
RL : right to left

Default value: #REQUIRED

Remarks: If the lines follow each other in some order other than top-to-bottom, bottom-to-top, left-to-right, or right-to-left, then some value other than those indicated should be used.

This attribute is required.

Example:

```
<writingSystemDeclaration lang=eng id='deu'
 name='-//TEI P2: 1993//WSD Modern German//en' date='1993-06-01'>
 <language iso639='deu'>New High German</language>
 <script>Latin script with diacritics.</script>
 <direction chars='LR' lines='TB'>
 <!-- ... -->
</writingSystemDeclaration>
```

If more than one direction is specified, it means that the script in question may be written in any of the directions named. In some cases, the directions may be mixed within a single document, in others not; no information is given on this point by the writing system declaration, as it is not usefully formalizable. Experts on the script must be consulted for details. The following example shows one way to declare that a script may be written in any horizontal or vertical direction.

```
<writingSystemDeclaration lang=eng id=jpn
 name='-//TEI P2: 1993//WSD JIS 0208//en' date='1993-06-01'>
 <language iso639='jpn'>Modern Japanese</language>
 <script>normal Japanese writing, with mixture of hiragana,
         katakana, and kanji.</script>
 <direction chars='LR' lines='TB'>
 <direction chars='RL' lines='TB'>
 <direction chars='LR' lines='BT'>
 <direction chars='RL' lines='BT'>
 <direction chars='TB' lines='LR'>
 <direction chars='BT' lines='LR'>
 <direction chars='TB' lines='RL'>
 <direction chars='BT' lines='RL'>
 <!-- Japanese can, in effect, be written in any direction. -->
 <!-- Individual documents may, but need not, mix the       -->
 <!-- various directions.                                   -->
 <!-- ... -->
</writingSystemDeclaration>
```

The following example shows another way to specify that a script may be written in any horizontal or vertical direction.

```
<writingSystemDeclaration lang=eng id=jpn
 name='//TEI P2: 1993//WSD JIS 0208//en' date='1993-06-01'>
 <language iso639='jpn'>Modern Japanese</language>
 <script>normal Japanese, with mixture of hiragana, katakana,
         and kanji.</script>
 <direction chars='LR RL' lines='TB BT'>
 <direction chars='TB BT' lines='LR RL'>
 <!-- Japanese can, in effect, be written in any direction. -->
 <!-- Individual documents may, but need not, mix the       -->
 <!-- various directions.                                   -->
 <!-- ... -->
</writingSystemDeclaration>
```

In some cases, the *lines* and *chars* attributes may need to take special values. Some scripts are written in “boustrophedon” (turning back and forth): i.e. one line is written left to right, the next right to left, and so on. Such a writing system might have its direction declared as shown:

```
<direction chars='boustrophedon:  LR, then RL, then LR, etc.'
           lines='TB'>
```

Remarks: This element describes conventional presentation; all scripts are subject to unusual treatment in special circumstances, and such unusual directions need not be described here. The treatment of numerals in Latin, Hebrew, and Arabic script, being well understood, need not be documented separately here.[1]
&@fn@sep

[1] Although the scripts run in opposite directions, they write numbers in the same direction; the usual view is that the numbers in Hebrew and Arabic run left to right, like those in Latin script, but it is also possible to claim that the numbers in

The **<direction>** element is informational only, not normative. It is intended to alert those responsible for implementing support for a given writing system to an essential fact of how it is written. If only a single direction is specified, it is safe to infer that the script may legitimately be presented in that direction. If multiple directions are specified, it may be legitimate to present text in any one of them, or it may be necessary to support multiple directions in display of a single document. If the latter, then changes of direction should be given using the global *rend* attribute.

If no direction element is given, the only safe assumption is that any direction is possible and experts must be consulted before attempting to implement support for the writing system.

Part: auxiliary tag set for writing system declarations.
Member of classes:
DTD file:
Data description: Empty.
May occur within: writingSystemDeclaration
May contain: EMPTY
Declarations:

```
<!ELEMENT direction     - O  EMPTY                            >
<!ATTLIST direction          %a.global;
          chars              CDATA            #REQUIRED
          lines              CDATA            #REQUIRED       >
```

Discussed in 27.3, “Describing the Writing System,” on page 7.
This tag is recommended.

<entitySet> (i.e. base entity set)

Description: identifies a public or private entity set whose mappings between entity names and characters are to be incorporated (perhaps with modifications) into this writing system declaration.
Attributes: [None: global and inherited attributes only.]
Example:

Remarks: Reference in a WSD to an entity set makes the set of entity-name-to-character mappings defined in the entity set available for use in text to which the WSD applies. Unless the character-to-entity-name mappings are modified by the **<exceptions>** element, any name in the entity set referred to may be used with its standard meaning in any text to which the WSD applies.

Latin scripts run right to left, like those in Arabic and Hebrew. There is no single satisfactory answer to this question. Fortunately, it is not necessary to answer it before creating WSDs for the scripts involved.

Since standard public entity sets are not always completely explicit about the distinctions among characters, glyphs, graphemes, and allographs, the TEI will provide standard writing system declarations which explicitly document the mappings provided by some commonly used public entity sets; it is recommended that similar documentation be provided for locally developed entity sets.

Part: auxiliary tag set for writing system declarations.
Member of classes: basestd
DTD file:
Data description: Empty.
May occur within: characters
May contain: EMPTY
Declarations:

```
<!ELEMENT entitySet     - O  EMPTY                          >
<!ATTLIST entitySet          %a.global;
                             %a.baseStandard;               >
```

Discussed in 27.4.1, “Base Components of the WSD,” on page 9.
This tag is optional.

<exceptions>

Description: documents ways in which a writing system declaration differs from the coded character sets, base writing system declarations, and entity sets which form its bases.
Attributes: [None: global and inherited attributes only.]
Example:

```
<writingSystemDeclaration
    name='-//TEI P2: 1993//WSD TLG Beta code//en'
    date='1993-06-01'>
<!-- ... -->
<characters>
    <baseWsd name='-//TEI P2: 1993//WSD ISO 646 IRV//en'
             authority='TEI'>
    <exceptions>
       <character> ... </character>
       <character> ... </character>
       <character> ... </character>
       <!-- ... -->
    </exceptions>
</characters>
```

Remarks:
Part: auxiliary tag set for writing system declarations.
Member of classes:
DTD file:

Data description: Contains a series of **<character>** elements, each documenting one character unit of the writing system.
May occur within: characters
May contain: character
Declarations:

```
<!ELEMENT exceptions    - O  (character*)                      >
<!ATTLIST exceptions         %a.global;                        >
```

Discussed in 27.4.2, “Exceptions in the WSD,” on page 11.
This tag is optional.

<extFigure> (i.e. external figure)

Description: (in a writing system declaration) refers to a figure or illustration depicting the character form, which is stored in some declared notation external to the text.
Attributes:

notation: identifies the notation in which the figure is stored.
Data type: NAME
Value: a valid SGML name associated with a given notation by means of an SGML NOTATION declaration in the document type definition.
Default value: #REQUIRED
Remarks:
This attribute is required.

entity: gives the SGML name of the external entity which contains the figure.
Data type: ENTITY
Value: a valid SGML name associated with the external entity by means of an SGML ENTITY declaration in the document type definition.
Default value: #REQUIRED
Remarks:
This attribute is required.

Example:

An image of the character form may be stored and transmitted in any desired graphics format. The declaration of the notation may specify a local system identifier for the processor (here it is imagined that we use a program called “pddraw.exe” to process files in TIFF format), thus:

```
<!-- in the DTD: -->
<!NOTATION TIFF system 'pddraw.exe'>
<!ENTITY lcthorn system 'lcthorn.TIF' NDATA TIFF>

<!-- in the WSD itself: -->
<extFigure notation=TIFF entity='lcthorn'>
```

The graphics format might also be declared with a public identifier in addition to the system identifier; this is likely to be more comprehensible and thus more useful to users of a different system:

```
<!-- in the DTD: -->
<!NOTATION TIFF
           PUBLIC '-//XXX//NOTATION Tagged Image File Format//EN'
           'pddraw.exe'>
<!ENTITY lcthorn system 'lcthorn.TIF' NDATA TIFF>

<!-- in the WSD itself: -->
<extFigure notation=TIFF entity='lcthorn'>
```

Character shapes may be conveniently conveyed in forms other than graphics images; one might, for example, define a character shape using a font-design program such as Donald Knuth's MetaFont program (which can be used to generate fonts for processing with TeX):

```
<!-- in the DTD: -->
<!NOTATION metafont
           PUBLIC '-//DEK//NOTATION MetaFont//EN'
           'mf.exe'>
<!ENTITY lcthorn system 'lcthorn.TIF' NDATA metafont>

<!-- in the WSD itself: -->
<extFigure notation=metafont entity='lcthorn'>
```

Remarks:
Part: auxiliary tag set for writing system declarations.
Member of classes:
DTD file:
Data description: Empty.
May occur within: form
May contain: EMPTY
Declarations:

```
<!ELEMENT extFigure     - O  EMPTY                          >
<!ATTLIST extFigure          %a.global;
          notation           NAME              #REQUIRED
          entity             ENTITY            #REQUIRED      >
```

Discussed in 27.4.2, “Exceptions in the WSD,” on page 11.
This tag is optional.

<figure>

Description: (in a writing system declaration) contains an image of a character form, stored in-line in some declared notation.
Attributes:

notation: identifies the notation in which the figure is encoded.
Data type: NAME
Value: a valid SGML name associated with a given notation by means of an SGML NOTATION declaration in the document type definition.
Default value: #REQUIRED
Remarks:
This attribute is required.

Example:

```
<!-- in the DTD -->
<!NOTATION charcell
           PUBLIC '-//Anonymous//NOTATION
                   8x14 character cell format//en' >

<!-- 'CHARCELL' is a simple-minded notation for transcribing  -->
<!-- character images in the form used by the old IBM PC:     -->
<!-- an eight-by-fourteen pixel character cell.  Each pixel   -->
<!-- is represented by a period (if off) or an X (if on).     -->
<!-- Whitespace and newlines are ignored, but for obvious     -->
<!-- reasons it's convenient to have a newline after each     -->
<!-- line of pixels.                                          -->

<!-- N.B.  This 'notation' is given as an example; it is not  -->
<!-- (repeat, NOT) recommended for serious use.               -->

<!-- in the WSD itself: -->
<character class='lexical'>
<desc>Latin lowercase letter thorn</desc>
<form entityStd='thorn' entityLoc='t'>
<figure notation=charcell>
 ........
 ........
 XXX.....
 .XX.....
 .XX.....
 .XXXXX..
 .XX..XX.
 .XX..XX.
 .XX..XX.
 .XXXXX..
 .XX.....
 .XX.....
 XXXX....
 ........
</figure>
</form>
</character>
```

Remarks: Inline storage of non-SGML data may be convenient for a variety of reasons, but complications occasionally ensue if any portion of the data can be interpreted by the SGML parser as ending some open SGML element. For this reason, inline storage should be used only for extremely simple notations such as the one shown, or by users well versed in dealing with SGML and graphics. In general, external storage of figures is recommended because less error prone.

Part: auxiliary tag set for writing system declarations.
Member of classes:
DTD file:
Data description: May contain non-SGML character data.
May occur within: form
May contain: CDATA
Declarations:

```
<!ELEMENT figure        - -  CDATA                              >
<!ATTLIST figure             %a.global;
          notation           NAME              #REQUIRED        >
```

Discussed in 27.4.2, “Exceptions in the WSD,” on page 11.
This tag is optional.

<form> (i.e. letter form)

Description: identifies one letter form taken by a particular character in a writing system declaration.
Attributes:

string: gives the byte string used to encode the letter form in the text.
Data type: CDATA
Value: any string of characters (often a single byte)
Default value: #IMPLIED
Example:

```
<form string='a/'>
   <desc>lowercase Greek alpha with acute accent</desc>
</form>
```

Remarks: If the character is encoded only using entity references, then the value of *string* should be '' (the empty string).

In coded character sets which use character-set shifting (e.g. JIS 0208), the *string* attribute should typically contain the required shift characters, in order to render the value unambiguous. In such a case, there is no expectation that every occurrence of the character will be immediately preceded by the shift sequence; processing software is responsible for understanding the shift mechanism and acting accordingly.

The same string value may not appear on more than one **<form>** elements (except the empty string), unless each occurrence is associated with a different coded character set.

This attribute is optional.

codedCharSet (i.e. coded character set): specifies which base coded character set the *string* value occurs in.
Data type: IDREF
Value: a reference to the SGML identifier of a **<codedCharSet>** element in the current writing system declaration.
Default value: #IMPLIED
Example:

Remarks: If more than one **<codedCharSet>** is specified as a base component of the writing system declaration, then it is expected that character-set shifting is in use, as described in ISO 2022 or some equivalent. In this case, each **<form>** element which has a value for the *string* attribute should also identify, by means of the *codedCharSet* attribute, which identifies which coded character set actually contains the string in question. Proper shifting among character sets is the responsibility of the user.
This attribute is optional.

entityStd (i.e. standard entity name): gives the name of one or more entities defined for this character form in some standard entity set(s).
Data type: ENTITIES
Value: One or more valid SGML entity names declared in the document type definition of the WSD; the entity must also be included in an entity set mentioned in an **<entitySet>** declaration in the current writing system declaration or in some base writing system referred to by a **<baseWsd>** element.
Default value: #IMPLIED
Example:

```
<form entityStd='thorn'>
  <desc>lowercase Old English/Icelandic thorn</desc>
<form>
```

Remarks: If the same letter form is defined by more than one public entity set, more than one value may appear in this attribute.

The same entity name may not appear in the *entityStd* or *entityLoc* attributes of more than one **<form>** element.
This attribute is optional.

entityLoc (i.e. local entity name): gives one or more entity names used locally for this character form.
Data type: ENTITIES
Value: One or more valid SGML entity names declared in the document type definition of the WSD; the entity must also be included in an entity set mentioned in an **<entitySet>** declaration in the current writing system declaration or in some base writing system referred to by a **<baseWsd>** element.
Default value: #IMPLIED
Example:

```
<form entityStd='thorn' entityLoc='t'>
  <desc>lowercase Old English/Icelandic thorn</desc>
  <note>The standard entity name is 'thorn'; the local entity 't'
        is used for brevity and legibility.</note>
<form>
```

Remarks: The same entity name may not appear in the *entityStd* or *entityLoc* attributes of more than one **<form>** element.
This attribute is optional.

ucs-4 (i.e. universal-character-set code): gives the position of the character form in the thirty-two bit “universal character set” defined by ISO 10646.
Data type: CDATA
Value: one or more sets of two or four two-digit hexadecimal numbers giving a valid ISO 10646 code point for the character form; for legibility the two-digit hexadecimal numbers should be separated by hyphens. If more than one UCS-4 code is associated with a given character form, the two UCS-4 codes should be given separated by blanks. If the character form is associated with a sequence of UCS-4 codes (e.g. a base character followed by one or more non-spacing diacritics), then the components of the sequence should be separated by '+'.
Default value: #IMPLIED
Example:

Remarks: The same UCS-4 code (or sequence) may not appear within more than one **<character>** element within the writing system declaration. It may however appear on several forms of the same character.

Multiple UCS-4 codes can be given for a single character; this allows sequences treated as distinct by ISO 10646 to be documented as referring to a single “character” as defined by the WSD (e.g. “lowercase a-umlaut” and “lowercase a” plus “umlaut”).

If a single UCS-4 code is to be treated as relating to two distinct “characters” as defined by the WSD (e.g. to reverse the effects of Han unification on some character), then one of the **<character>** elements should be associated with the UCS-4 code in the normal way, and the others should call attention to the relevant UCS-4 code by a comment in a **<note>** element.
This attribute is optional.

afiicode (i.e. AFII code): gives one or more codes associated with this letter form by the Association for Font Information Interchange.
Data type: CDATA
Value: any valid AFII identifier.
Default value: #IMPLIED
Example:

Remarks: The AFII tables are designed as an inventory of *glyphs* (identifiably distinct shapes, leaving differences of font design out of account—one character may be associated with several glyphs, and each glyph with items in several different fonts). Because the same glyph may be associated with more than one character (in some fonts, for example, the lowercase letter L and the digit 1 share the same glyph), the value of *afiiCode* is used for informational purposes only and need not be unique within the writing system declaration.

This attribute is optional.

Example:

Remarks: The **<form>** element documents one form of a character; in most cases, there will be only one. If more than one form is given, in general, they are to be regarded as free variants of the character unless otherwise specified in the notes.

The distinction between **<character>** and **<form>** makes it possible to distinguish, in an encoding, among different letter forms (which may have historical, aesthetic, linguistic, or other significance) without having to claim that the different forms constitute different "characters" in any normal sense. (Using the technical terms occasionally encountered, the **<form>** element can be used to record each *allograph* of a given character or *grapheme*.) The concepts of "character" and "letter form", however, vary from analyst to analyst; the decision to treat a given set of forms as a single character or as a set of characters is not always obvious, and may require the application of considerable learning and judgement. The **<note>** element should be used to record the reasoning behind any particularly difficult decision.

Part: auxiliary tag set for writing system declarations.

Member of classes:

DTD file:

Data description: May contain a series of description element, optionally one or more figure elements showing the character form in question, and optionally a series of notes.

May occur within: character

May contain: desc extFigure figure note

Declarations:

```
<!ELEMENT form          - O  (desc+, (figure | extFigure)*,
                              note*)                           >
<!ATTLIST form               %a.global;
          string             CDATA            #IMPLIED
          codedCharSet       IDREF            #IMPLIED
          entityStd          ENTITIES         #IMPLIED
          entityLoc          ENTITIES         #IMPLIED
          ucs-4              CDATA            #IMPLIED
          afiicode           CDATA            #IMPLIED       >
```

Discussed in 27.4.2, "Exceptions in the WSD," on page 11.

This tag is mandatory when applicable.

Class Name: **global**
Description: defines global attributes for the writing system declaration.
Attributes:

id: gives a unique identifier for the element
Data type: ID
Value: any valid SGML name unique within the document.
Default value: #IMPLIED
Remarks:
This attribute is optional.

lang (i.e. language): gives the language in which the content of the element is written.
Data type: CDATA
Value: Should be a language code from ISO 639.
Default value: %INHERITED
Remarks: This attribute functions like the global attribute of the same name in the main TEI DTD; for technical reasons it is declared differently.
This attribute is optional.

Remarks:
Part: auxiliary tag set for writing system declarations.
Member of classes:
DTD file: teiwsd2
Discussed in 27.1, “Overall Structure of Writing System Declaration,” on page 3.

Entity Name: **INHERITED**
Description: as a default attribute value, indicates that if not specified, the attribute value is inherited from corresponding attribute of the parent element.
Remarks: For the SGML parser, this entity has the same effect as specifying a default value as “#IMPLIED”. For the user and the application program, however, “%INHERITED” has a different effect, specifying as it does exactly how the application program is to infer the correct default value: the value is to be inherited from the attribute of the same name on the immediately enclosing element. If that element, too, specifies no value, then its value will have been inherited from its own immediately enclosing element, etc. (If the attribute is not declared for all elements, the value is inherited from the nearest ancestor for which the attribute is declared.)

If no ancestor element has a value specified for the attribute, the value is undefined. Encoders are encouraged to provide an explicit value for inherited attributes on the outermost elements for which they are declared; it is, however, not an error for the outermost element to specify no attribute value for an attribute with a default of “%INHERITED”.

The most prominent example of attribute value inheritance is the TEI global attribute *lang*.
Expansion: '#IMPLIED'
Discussed in 3.7, “Element Classes,” [in separate fascicle].

Entity Name: **ISO-date**

Description: as the declared value of an attribute, indicates that the attribute value should be a legal ISO date in the format defined by ISO 714[?], viz. yyyy-mm-dd.

Remarks: For the SGML parser, this entity has the same effect as specifying a declared value of “CDATA”. For the user and the application program, however, “%ISO-date” documents a restriction which SGML is not currently capable of enforcing.

The most prominent examples of this declared value type are the *value* attribute of the **<date>** element in the core tag set, and the *date* attribute of the **<admin>** element in the base tag set for terminology.

Expansion: 'CDATA'

Discussed in 3.8.3, “Parameter Entities for TEI Keywords,” [in separate fascicle].

<language>

Description: identifies the language being described in the writing system declaration.

Attributes:

iso639: gives the standard language code from ISO 639.
Data type: CDATA
Value: any two- or three-letter code included included in ISO 639; if the language is not included in the list in ISO 639, the value should be given as '' (the empty string).
Default value: #REQUIRED
Example:

Remarks:
This attribute is required.

Example:

Remarks: In general, the “language” associated with a given writing system declaration will be a natural language; it may however be a dialect, an artificial language, or some other semiotic system conveniently treated as a language.

The content of the **<language>** element is a description in prose of what language the WSD describes. Usually this will simply be the conventional name of the language; more information may however be included as needed.

Specialized writing system declarations which document a public character set or entity set suitable for encoding several languages, and which are intended for use as a base component within other language-specific writing system declarations, should identify the **<language>** to which they apply as “Various”. Such writing systems should *not* be used directly by encoded texts; they should be named only in the **<baseWSD>** element of language-specific writing system declarations.

Part: auxiliary tag set for writing system declarations.
Member of classes:
DTD file:
Data description: May contain character data only.
May occur within: writingSystemDeclaration
May contain: #PCDATA
Declarations:

```
<!ELEMENT language      - O  (#PCDATA)                          >
<!ATTLIST language           %a.global;
          iso639             CDATA              #REQUIRED      >
```

Discussed in 27.2, “Identifying the Language,” on page 6.
This tag is required.

<note>

Description: (in a writing system) contains a note of any type.
Attributes: [None: global and inherited attributes only.]
Example:

```
<note>This writing system declaration describes the ad hoc
 character set developed for encoding Hebrew and Arabic for the
 Geniza Project at Princeton University.  It is intended for
 documentation of existing files only and should not be used to
 guide the implementation of new systems.
 </note>
```

A note may also relate to a single character:

```
<note>This character is the Japanese form of the unified Han
 character xxxx in ISO 10646; for the Korean form, use the
 preceding character.</note>
```

Remarks:
Part: auxiliary tag set for writing system declarations.
Member of classes:
DTD file:
Data description: May contain character data only.
May occur within: character form writingSystemDeclaration
May contain: #PCDATA
Declarations:

```
<!ELEMENT note          - O  (#PCDATA)                          >
<!ATTLIST note               %a.global;                         >
```

Discussed in 27.5, “Notes in the WSD,” on page 17.
This tag is optional.

<script>

Description: contains a prose description of the script declared by a writing system declaration.
Attributes: [None: global and inherited attributes only.]
Example:

```
<writingSystemDeclaration lang=eng id='iso8859-1'
    name='-//TEI P2: 1993//WSD ISO 8859-1: 1993//en'
    date='1993-06-01'>
 <language iso639=''>Various</language>
 <script>Latin script with diacritics.</script>
 <!-- ... -->
</writingSystemDeclaration>

<writingSystemDeclaration lang=eng id=jpn
 name='-//TEI P2: 1993//WSD JIS 0208//en' date='1993-06-01'>
 <language iso639='jpn'>Modern Japanese</language>
 <script>normal Japanese, with mixture of hiragana, katakana,
         and kanji.</script>
 <!-- ... -->
</writingSystemDeclaration>
```

Remarks: This element is provided for the sake of clarity and readability. Strictly speaking, it is almost always redundant, as knowledgeable readers will know what script is involved as soon as they examine the base character set or the character repertoire.
Part: auxiliary tag set for writing system declarations.
Member of classes:
DTD file:
Data description: May contain character data only.
May occur within: writingSystemDeclaration
May contain: #PCDATA
Declarations:

```
<!ELEMENT script        - O  (#PCDATA)                          >
<!ATTLIST script             %a.global;                         >
```

Discussed in 27.3, “Describing the Writing System,” on page 7.
This tag is required.

Entity Name: **TEI.wsdNames**

Description: file containing parameter entity declarations for all generic identifiers used in the writing system declaration
Remarks: The parameter entities in this file all take the same form as the two shown below:

```
<!ENTITY % n.div0 'div0' >
<!ENTITY % n.div1 'div1' >
```

Element and attribute-list declarations in the DTDs refer to the parameter entity *n.div1*, not directly to the generic identifier *div1*. As a result, the declarations will function as desired even if a new generic identifier is substituted (e.g. “caput” for “div1” and “liber” for “div0”:

```
<!ENTITY % n.div0 'liber' >
<!ENTITY % n.div1 'caput' >
```

This allows generic identifiers to be renamed conveniently, to provide names in languages other than English, or to provide shorter names than those documented here. See further chapter 31, “Modifying TEI DTDs,” [in separate fascicle].

Expansion: system 'wdgis2.ent'

Discussed in 27.1, “Overall Structure of Writing System Declaration,” on page 3.

<writingSystemDeclaration>

Description: declares the coded character set, transliteration scheme, or entity set used to transcribe a given writing system of a given language.

Attributes:

name: gives a formal name for the writing system declaration
Data type: CDATA
Value: any string of characters
Default value: #REQUIRED
Example:

```
<writingSystemDeclaration
    name='-//TEI P2: 1993//WSD ISO 646 IRV//en'
    date='1993-06-30'>
```

Remarks:
This attribute is required.

date: gives the date on which the writing system declaration was last revised.
Data type: %ISO-date
Value: A date in valid ISO format.
Default value: #REQUIRED
Example:

Remarks:
This attribute is required.

Example:

```
<writingSystemDeclaration id=eng lang=eng
  name='-//TEI P2: 1993//WSD Modern English//en'
  date='1993-05-25'>
  <language>Modern English</>
  <direction lines=TB chars=LR>
  <base name='ANSI X3.4' authority='national'>
  <exceptions></>
</writingSystemDeclaration>
```

Remarks:
Part: auxiliary tag set for writing system declarations.
Member of classes:
DTD file:
Data description: May contain only a prescribed series of sub-elements.
May occur within:
May contain: characters direction language note script
Declarations:

```
<!ELEMENT writingSystemDeclaration
                        - -  (language, script, direction*,
                             characters, note*)                >
<!ATTLIST writingSystemDeclaration
                             %a.global;
          name               CDATA               #REQUIRED
          date               %ISO-date           #REQUIRED      >
```

Discussed in 27.1, “Overall Structure of Writing System Declaration,” on page 3.
This tag is required.

Usability Report and Comment

If you have comments of any kind on the current draft of the TEI Guidelines, please complete this form and return it to one of the following addresses:

C. M. Sperberg-McQueen, Computer Center (M/C 135), University of Illinois at Chicago, 1940 W. Taylor Room 124, Chicago IL 60612-7352, USA. Bitnet: U35395@UICVM, Internet: U35395@uicvm.uic.edu

Lou Burnard, Oxford University Computing Service, 13 Banbury Road, Oxford OX2 6NN, UK. Fax: +44 (865) 273275, Janet: LOU@UK.AC.OXFORD.VAX Internet: LOU@vax.oxford.ac.uk

Further copies of this form may be downloaded from LISTSERV @ UICVM (or listserv @ uicvm.uic.edu). Send a mail message containing the line 'GET TEIURC MEMO TEI-L'

Your name:
Your postal address:

Your e-mail address (if any):
Your occupation:
Your academic background:
Your immediate reactions to the Guidelines (please tick or cross one only):

Relevance to your present concerns or interests: high medium low
Importance to your future research plans: high medium low
Comprehensibility/usability of current draft: high medium low

Detailed technical comment: [Please feel free to comment on any part of the Guidelines or in general. Comments relating to specific parts of the text are most helpful if they include a section or chapter number]

Thank you!

Temporary Index