3 Structure of the TEI Document Type Definition

This chapter describes the overall structure of the encoding scheme defined by these Guidelines. It introduces the conceptual framework within which the following chapters are to be understood, and describes the technical means by which that conceptual framework is implemented. It assumes familiarity with SGML or XML; see chapter 2 *A Gentle Introduction to XML*.

The TEI encoding scheme consists of a number of modules or *DTD fragments* which we refer to below as *tag sets*. Selected tag sets may be combined in many different ways, according to principles described in this chapter, within the framework of the TEI *main DTD*. Auxiliary tag sets are also defined for specific purposes independent of the TEI main DTD.

The DTD fragments from which the main TEI DTD is constructed may be classified as follows:

- core DTD fragments
- base DTD fragments
- additional DTD fragments

The first two sections of this chapter discuss these distinctions and list the specific tag sets included in each category. Section 3.3 *Invocation of the TEI DTD* describes how to invoke the TEI document type declaration, and how to specify which of the various base tag sets and optional additional tag sets are used in a document.

The *global attributes*, characteristics postulated of every element or tag in the encoding scheme, are defined in section 3.5 *Global Attributes*.

The remainder of the chapter contains a more technical description of the mechanisms used to implement the encoding scheme. It may be skipped at a first reading, but a proper understanding of the topics addressed here is essential for anyone planning to modify or extend the TEI encoding scheme in any way (see also chapter 29 *Modifying and Customizing the TEI DTD*), and also highly desirable for those wishing to take full advantage of its modular nature. The structure of the main TEI DTD file itself is outlined in section 3.6 *The TEI2.DTD File*. The *element classes* used to define smaller groups of elements and their characteristics are described in section 3.7 *Element Classes*. Both global attributes and element classes are implemented using *parameter entities*; various other uses of parameter entities in the TEI DTDs are discussed in section 3.8 *Other Parameter Entities in TEI DTDs*.

3.1 Main and Auxiliary DTDs

These Guidelines define a large number of elements for marking up documents, all of which are formally defined within the *document type declaration* (*DTD*) files provided by the TEI and documented in the remainder of the present document. They are grouped into *element sets* (also known as *tag sets* or *DTD fragments*), each comprising a set of declarations for elements which belong together in some respect, typically related to their intended application area.

All elements used to transcribe documents are available for use within the *main DTD* of the TEI and are defined in Parts III and IV of these Guidelines. There are DTD fragments for prose and mixed matter, verse and verse collections, drama, dictionaries, analysis and interpretation of text, text criticism, etc. A full list, including the files in which they are defined, and the rules determining their selection and combination, is given in section 3.2 *Core, Base, and Additional Tag Sets*.

A number of *auxiliary DTDs* are also defined in these Guidelines. These are used for the encoding of ancillary descriptive information useful when processing electronic documents. Part V of these Guidelines describes several such auxiliary document types, specifically:

independent header for use with sets of TEI headers regarded as documents in their own right, for example by libraries or archives exchanging details of their holdings (see chapters 5 *The TEI Header* and 24 *The Independent Header*).

writing system declaration used to define and document character sets or transliteration schemes (see chapters 4 *Languages and Character Sets* and 25 *Writing System Declaration*).

feature system declaration used to define and document sets of analytic features (see chapters 16 *Feature Structures* and 26 *Feature System Declaration*).

tag set declaration used to define and document descriptive documentation for TEI-conformant tag sets (see chapter 27 *Tag Set Documentation*).

An independent header typically describes the encoding of a specific document, but in the case of a planned corpus or collection, it may define a set of encoding practices common to all texts in the collection. The other auxiliary document types provide information likely to be relevant to many documents, rather than to individual documents.

When individual TEI documents are exchanged between sites, they should be accompanied by whatever auxiliary documents apply to them. When larger groups of documents are exchanged, the relevant auxiliary documents need be exchanged only once. For further information see chapter 30 *Rules for Interchange*.

The DTD files containing these auxiliary DTDs are:

teishd2.dtd independent header

teiwsd2.dtd writing system declaration

teifsd2.dtd feature system declaration

teitsd2.dtd tag set declaration

Some of these auxiliary DTDs also make use of the core tag set defined as part of the main TEI DTD; this is described in the relevant chapters of part V.

3.2 Core, Base, and Additional Tag Sets

The main TEI DTD is constructed by selecting an appropriate combination of smaller tag sets, each containing some set of tags likely to be used together. These building blocks include:

core tag sets standard components of the TEI main DTD in all its forms; these are always included without any special action by the encoder;

base tag sets basic building blocks for specific text types; exactly one base must be selected by the encoder (unless one of the 'combined' bases is used);

additional tag sets extra tags useful for particular purposes. All additional tag sets are compatible with all bases and with each other; an encoder may therefore add them to the selected base in any combination desired.

Each tag set is contained in one or more system files, which are defined by appropriate *parameter entity declarations* and invoked as a unit by *parameter entity references*.[34] Several such declarations may be needed to invoke all parts of a given tag set, since as well as defining elements or attributes, a tag set may (for example) add new items to the set of global attributes or add classes to the system of element classes. Consistent naming principles are applied throughout the TEI scheme for these and other entities. Thus, assuming a tag set named xxx, the following parameter entities may be encountered:

TEI.xxx used to enable or disable tag set xxx; must have the value `INCLUDE` (tag set is enabled) or, by default, `IGNORE` (tag set not enabled).

TEI.xxx.ent refers to a system file containing any parameter entity declarations unique to tag set xxx.

TEI.xxx.dtd refers to a system file containing the element and attribute list declarations for tag set xxx.

[34] A *parameter entity* is an entity used only in markup declarations; references to parameter entities are delimited by a percent sign and a semicolon rather than the ampersand and colon used for *general entity* references. The entity TEI.core.ent, for example, would be referred to using the string `%TEI.core.ent;`. Parameter entities can also be used to control the inclusion or exclusion of *marked sections* of the document or DTD; the TEI DTD uses marked sections to handle the selection of different base and additional tag sets.

a.xxx contains definitions of attributes which are to be added to the set of global attributes when tag set xxx is enabled.

m.comp.xxx a list of any component-level elements unique to base tag set xxx (for a definition of component-level elements, see section 3.7 *Element Classes*).

mix.xxx a special entity for use in defining the set of component-level elements when the mixed base tag set is in use.

gen.xxx a special entity for use in defining the set of component-level elements when the general base tag set is in use.

Few tag sets declare all of these entities; only those actually used are declared.

The interpretation of the parameter entity declarations, and the inclusion of the appropriate tag sets, are handled by a single 'driver file' for the main TEI DTD. This file, tei2.dtd, is described in detail below in section 3.6 *The TEI2.DTD File*. The remainder of the present section identifies the files in which each tag set is contained, and the parameter entities associated with them.

3.2.1 The Core Tag Sets

Two 'core' tag sets are always included in every invocation of the main TEI DTD. The tags and attributes that they contain are therefore available to any TEI document. The parameter entities used for this purpose, and the files they refer to, are:

TEI.core.dtd refers to the file teicore2.dtd, which declares the core tags defined in chapter 6 *Elements Available in All TEI Documents*

TEI.header.dtd refers to the file teihdr2.dtd, which declares the tags of the TEI header defined in chapter 5 *The TEI Header*

Together with these tag sets, part II also documents a tag set for default text structure and front and back matter. This tag set is embedded by the base tag set selected, and may vary with the base; it is therefore described in the next section.

3.2.2 The Base Tag Sets

The base tag sets are those which define the basic building blocks of different text types. The basic structures of verse (line, stanza, canto, etc.), for example, are not those of prose (paragraph, section, chapter, etc.), while dictionaries use yet another set of basic structures. Each base corresponds to one chapter of Part III of this document.

In general, exactly one base tag set must be selected for any TEI-conformant document. Errors will result if none, or more than one, is selected, because the same elements may be differently defined in different base tag sets. For documents which mingle structurally dissimilar elements and require elements from more than one base, however, either the *mixed base* or the *general base* may be used; see section 3.4 *Combining TEI Base Tag Sets*. These bases require the encoder to specify which of the other bases are to be combined.

The encoder selects a base tag set by declaring the appropriate parameter entity with the replacement text INCLUDE. To invoke the base tag set for prose, for example, the encoder must ensure that the DTD subset in the document contains the declaration:

```
<!ENTITY % TEI.prose 'INCLUDE' >
```

The entities used to select the different base tag sets, and the files containing the declarations for each base, are listed below.

TEI.prose selects the base tag set for prose, contained in teipros2.dtd.

TEI.verse selects the base tag set for verse, contained in teivers2.dtd and teivers2.ent.

TEI.drama selects the base tag set for drama, contained in teidram2.dtd and teidram2.ent.

TEI.spoken selects the base tag set for transcriptions of spoken texts, contained in teispok2.dtd and teispok2.ent.

TEI.dictionaries selects the base tag set for print dictionaries, contained in teidict2.dtd and teidict2.ent.

TEI.terminology selects the base tag set for terminological data files, contained in teiterm2.dtd, teiterm2.ent, teite2n.dtd, and teite2f.ent.

TEI.general selects the generic mixed-mode base base tag set, contained in teigen2.dtd.

TEI.mixed selects the base tag set for free mixed-mode texts, contained in teimix2.dtd.

As shown in the list, each base tag set is normally contained in one or two system files: a required one (with the extension 'dtd') defining the elements in the tag set and their attributes, and an optional one (with the file extension 'ent') defining any global attributes or specialized element classes enabled by that tag set. The parameter entities for these files have the same name as the enabling parameter entity for the base, with the suffixes 'ent' and 'dtd' respectively: the prose base, for example, is enabled by declaring the parameter entity TEI.prose as INCLUDE; this in turn enables declarations of TEI.prose.ent and TEI.prose.dtd as the system files teipros2.ent and teipros2.dtd. For further details, see section 3.6 *The TEI2.DTD File*.

Most base tag sets (but not necessarily all) embed common definitions of text structure, front matter, and back matter, by referring to three standard parameter entities; these are:

TEI.structure.dtd refers to the file teistr2.dtd, with default definitions for <text>, , etc.

TEI.front.dtd refers to the file teifron2.dtd, with tags for front matter

TEI.back.dtd refers to the file teiback2.dtd, with tags for back matter

These default-structure tags are documented in chapter 7 *Default Text Structure*.

3.2.3 The Additional Tag Sets

The additional tag sets define optional tags required by different encoders for different types of analysis and processing; each corresponds to a chapter in part IV of this document. In any TEI encoding, any or all of these additional tag sets may be made available, as they are all compatible with each other and with every base tag set. They are invoked in the same way as base tag sets, by defining the appropriate parameter entity as INCLUDE; the relevant parameter entities, and the files containing the additional tag sets, are these:

TEI.linking embeds the files teilink2.dtd and teilink2.ent, with tags for linking, segmentation, and alignment (chapter 14 *Linking, Segmentation, and Alignment*)

TEI.analysis embeds the files teiana2.dtd and teiana2.ent, with tags for simple analytic mechanisms (chapter 15 *Simple Analytic Mechanisms*)

TEI.fs embeds the file teifs2.dtd, with tags for feature structure analysis (chapter 16 *Feature Structures*)

TEI.certainty embeds the file teicert2.dtd, with tags for indicating uncertainty and probability in the markup (chapter 17 *Certainty and Responsibility*)

TEI.transcr embeds the files teitran2.dtd and teitran2.ent, with tags for manuscripts, analytic bibliography, and transcription of primary sources (chapter 18 *Transcription of Primary Sources*)

TEI.textcrit embeds the files teitc2.dtd and teitc2.ent, with tags for critical editions (chapter 19 *Critical Apparatus*)

TEI.names.dates embeds the files teind2.dtd and teind2.ent, with specialized tags for names and dates (chapter 20 *Names and Dates*)

TEI.nets embeds the file teinet2.dtd, with tags for graphs, digraphs, trees, and other networks (chapter 21 *Graphs, Networks, and Trees*) — not to be confused with the graphics markup of TEI.figures

TEI.figures embeds the files teifig2.dtd and teifig2.ent, with tags for graphics, figures, illustrations, tables, and formulae (chapter 22 *Tables, Formulae, and Graphics*) — not to be confused with the graph-theoretic markup of TEI.nets

TEI.corpus embeds the file teicorp2.dtd, with tags for additional tags for language corpora (chapter 23 *Language Corpora*)

Like the base tag sets, the additional tag sets are each contained in one or two system files: a required one (with the file extension 'dtd') defining the elements in the tag set and their attributes, and an optional one (with the file extension 'ent') defining any global attributes or specialized element classes enabled by that tag set. The parameter entities for these files have the same name as the enabling parameter entity for the tag set, with the suffixes 'ent' and 'dtd' respectively: the additional tag set for linking, segmentation, and alignment, for example, is enabled by declaring the parameter entity TEI.linking as INCLUDE; this in turn enables declarations of TEI.linking.ent and TEI.linking.dtd as the system files teilink2.ent and teilink2.dtd.

3.2.4 User-Defined Tag Sets

As described in chapter 29 *Modifying and Customizing the TEI DTD*, users may modify the markup language defined here by renaming elements, suppressing elements, adding new elements, or modifying element or attribute-list declarations. In general, local modifications will be most conveniently grouped into two files: one containing the local modifications to parameter entities used in the DTDs, and the other containing new or modified declarations of elements and their attributes. These files will be embedded in the TEI DTD if they are associated with the following two parameter entities:

TEI.extensions.ent local modifications to parameter entities

TEI.extensions.dtd declarations of new elements and modified declarations for existing elements

In some cases, users may wish to provide completely new base or additional tag sets, to be invoked in the same way as those defined in this document; such tag sets should also be divided into 'entity files' and 'DTD files' in the same way as the standard tag sets. Such modifications should be undertaken only with a thorough understanding of the interface among core, base, and additional tag sets as documented in the final sections of this chapter; see in particular section 3.6.2 *Embedding Local Modifications*.

Further recommendations for the creation of user-defined extension or modification are provided in chapters 29 *Modifying and Customizing the TEI DTD* and 28 *Conformance*.

3.3 Invocation of the TEI DTD

A TEI SGML document must begin with a document type definition (DTD), as must a valid TEI XML document (though not a merely well-formed TEI XML document). Local systems may allow the DTD to be implicit, but for interchange purposes it *must* be explicit for both SGML and XML. Because of its highly modular nature, it may in any case be desirable for the component parts of the TEI DTD to be made explicit even for local processing.

The simplest version of the TEI DTD names the main TEI DTD file as an external file, and specifies a single base tag set for use in the document, using the parameter entity names specified in section 3.2 *Core, Base, and Additional Tag Sets*. For example, a document using the base tag set for prose will begin with a document type declaration something like this:

```
<!DOCTYPE TEI.2 PUBLIC "-//TEI P4//DTD Main Document Type//EN" "tei2.dtd" [
      <!ENTITY % TEI.XML   'INCLUDE' >
      <!ENTITY % TEI.prose 'INCLUDE' >
]>
```

A document using the base tag set for drama will define a different parameter entity:

```
<!DOCTYPE TEI.2 PUBLIC "-//TEI P4//DTD Main Document Type//EN" "tei2.dtd" [
      <!ENTITY % TEI.XML   'INCLUDE' >
      <!ENTITY % TEI.drama 'INCLUDE' >
]>
```

If one or more of the *additional tag sets* described in Part IV are to be used, they are invoked in the same way as the base tag set. A document using the base tag set for prose, with the additional tag sets for text criticism and for linking, segmentation, and alignment, for example, will begin with a document type declaration something like this:

```
<!DOCTYPE TEI.2 PUBLIC "-//TEI P4//DTD Main Document Type//EN" "tei2.dtd" [
      <!ENTITY % TEI.XML      'INCLUDE' >
```

```
<!-- TEI base tag set specified here: ... -->
      <!ENTITY % TEI.prose 'INCLUDE' >

<!-- TEI additional tag sets optionally specified here: ... -->
      <!ENTITY % TEI.textcrit 'INCLUDE' >
      <!ENTITY % TEI.linking 'INCLUDE' >
 ]>
```

If local modifications are used, they may be stored in separate files and pointed to using the parameter entities TEI.extensions.ent and TEI.extensions.dtd. If such local modifications are added to the example just given, this is the result:

```
<!DOCTYPE TEI.2 PUBLIC "-//TEI P4//DTD Main Document Type//EN" "tei2.dtd" [
      <!ENTITY % TEI.XML      'INCLUDE' >

<!-- Local modifications to the TEI DTD declared here.  They
      will be embedded at an appropriate point in the main
      DTD. ... -->
      <!ENTITY % TEI.extensions.ent SYSTEM 'project.ent' >
      <!ENTITY % TEI.extensions.dtd SYSTEM 'project.dtd' >

<!-- TEI base tag set specified here: ... -->
      <!ENTITY % TEI.prose 'INCLUDE' >

<!-- TEI additional tag sets specified here: ... -->
      <!ENTITY % TEI.textcrit 'INCLUDE' >
      <!ENTITY % TEI.linking 'INCLUDE' >
 ]>
```

If the document requires tags which are defined in different base tag sets (e.g. prose and drama) or embeds smaller texts which use different base tag sets, then one of the mixed-type bases must be used. Their proper invocation is described below in section 3.4 *Combining TEI Base Tag Sets*.

3.4 Combining TEI Base Tag Sets

The TEI DTD has been designed to simplify the task of choosing an appropriate set of tags for the text in hand. The core tag set includes tags appropriate to the majority of simple tagging requirements for prose, verse, and drama, irrespective of the base tag set chosen. For more detailed tagging, the encoder may choose the prose base for prose texts, the verse base for verse, and so on.

In discussing these base tag sets elsewhere in these Guidelines, it is generally assumed for clarity of exposition that a text will fall into one, not several, of these types. It is not uncommon, however, for a text to combine prose and verse, or other forms treated by the TEI as different bases. Examples include:

- when the text is a collection of other texts, which do not all use the same base: e.g. an anthology of prose, verse, and drama
- when the text contains other smaller, embedded texts: e.g. a poem or song included in a prose narrative
- when some sections of the text are written in one form, and others in a different form: e.g. a novel where some chapters are in prose, others take the form of dictionary entries, and still others the form of scenes in a play
- when the text moves back and forth among forms not between sections but within a single section: e.g. mixed prose-and-verse forms like many pastorals or like some portions of the Poetic Edda

The TEI DTD provides the following mechanisms to handle these cases:

- a definition of a corpus or collection as a series of <TEI.2> documents, sharing a common TEI header (see chapter 23 *Language Corpora*)
- a definition of composite texts which comprise front matter, a group or several possibly nested groups of collected texts, themselves possibly composite (see section 7.3 *Groups of Texts*)

- a notion of *embedded text* which allows one text to be embedded within another (that is, <text> is defined as a component-level element, as described briefly at the conclusion of section 7.3 *Groups of Texts*)

Whichever mechanism is adopted, if the whole of the resulting document is to be parseable by the main TEI DTD it may need to combine elements from different TEI base tag sets. Two special-purpose base tag sets are defined for this purpose:

- the *general* base, which allows different sections of a text to use different bases, but ensures that each section uses only one base
- the *mixed* base, which allows chunk- and inter-level elements from any base to mix within any text division

When either of these 'combined' bases is used, the user must specify all of the other bases to be included in the mix as well as either the general or the mixed base. This is the only exception to the general rule that no more than one base tag set may be enabled in a TEI document. The following set of declarations for example allows for any mixture of the low level structural tags defined in the prose, drama and dictionary base tag sets:

```
<!DOCTYPE TEI.2 PUBLIC "-//TEI P4//DTD Main Document Type//EN" "tei2.dtd" [
   <!ENTITY % TEI.XML   'INCLUDE' >
   <!ENTITY % TEI.mixed 'INCLUDE' >
   <!ENTITY % TEI.prose 'INCLUDE' >
   <!ENTITY % TEI.drama 'INCLUDE' >
   <!ENTITY % TEI.dictionaries 'INCLUDE' >
<!-- Structurally, Moby Dick is not your
    everyday common or garden variety novel ... -->
]>
```

The following set of declarations has the same effect, but with the additional restriction that each text division (i.e. each member of the element class divn) must be homogenous with respect to the mixture of available bases. Because in a 'general' base, each of the text may use a different base, the divisions of the text prefixed by this set of declarations will each be composed of elements taken solely from one of the prose, verse, or dictionary base tag sets:

```
<!DOCTYPE TEI.2 PUBLIC "-//TEI P4//DTD Main Document Type//EN" "tei2.dtd" [
   <!ENTITY % TEI.XML      'INCLUDE' >
   <!ENTITY % TEI.general  'INCLUDE' >
   <!ENTITY % TEI.prose    'INCLUDE' >
   <!ENTITY % TEI.drama    'INCLUDE' >
   <!ENTITY % TEI.dictionaries 'INCLUDE' >
]>
```

The actual DTD fragments for the combined bases do nothing but embed the default tag set for overall text structure. The mixed-base tag set is in file teimix2.dtd:

```
<!-- 3.4: Mixed-Base Tag Set-->
<!--Text Encoding Initiative Consortium:
Guidelines for Electronic Text Encoding and Interchange.
Document TEI P4, 2002.
Copyright (c) 2002 TEI Consortium. Permission to copy in any form
is granted, provided this notice is included in all copies.
These materials may not be altered; modifications to these DTDs should
be performed only as specified by the Guidelines, for example in the
chapter entitled 'Modifying the TEI DTD'
These materials are subject to revision by the TEI Consortium. Current versions
are available from the Consortium website at http://www.tei-c.org-->
<!ENTITY % TEI.structure.dtd PUBLIC '-//TEI P4//ELEMENTS Default Text
Structure//EN' 'teistr2.dtd' >%TEI.structure.dtd;
<!-- end of 3.4-->
```

The general-base tag set is in file teigen2.dtd:

```
<!-- 3.4: General-Base Tag Set-->
<!--Text Encoding Initiative Consortium:
Guidelines for Electronic Text Encoding and Interchange.
```

```
Document TEI P4, 2002.
Copyright (c) 2002 TEI Consortium. Permission to copy in any form
is granted, provided this notice is included in all copies.
These materials may not be altered; modifications to these DTDs should
be performed only as specified by the Guidelines, for example in the
chapter entitled 'Modifying the TEI DTD'
These materials are subject to revision by the TEI Consortium. Current versions
are available from the Consortium website at http://www.tei-c.org-->
<!ENTITY % TEI.structure.dtd PUBLIC '-//TEI P4//ELEMENTS Default Text
Structure//EN' 'teistr2.dtd' >%TEI.structure.dtd;
<!-- end of 3.4-->
```

Although these two fragments are identical, they define two different bases, because of the way their component level elements are used, as further described in 3.7.8 *Components in Mixed and General Bases* below.

3.5 Global Attributes

The following attributes are defined for every TEI element.[35]

id provides a unique identifier for the element bearing the ID value.

n gives a number (or other label) for an element, which is not necessarily unique within the document.

lang indicates the language of the element content, usually using a two- or three-letter code from ISO 639.

rend indicates how the element in question was rendered or presented in the source text.

Some tag sets (e.g. those for terminology, linking, and analysis) define other global attributes; these are documented in the appropriate chapters of Part III and Part IV. See also section 3.7.1 *Classes Which Share Attributes*.

An additional attribute, TEIform, is also defined for every TEI element. Unlike the other attributes defined for every element, TEIform is not defined by class global because its default value is different in every case and must be defined individually for each element.[36]

TEIform indicates the standard TEI name (generic identifier) for a given element.

Any TEI element may be given values for id, n, lang, rend, or TEIform, simply by specifying values for these attributes. The following two examples convey the same information about the text: that the material transcribed occurs within a <p> element (paragraph). They differ only in that the second provides an identifier for the paragraph, to which other elements (e.g. notes or hypertext links) can conveniently refer.

```
<p>If to do were as easy as to know what were
good to do, chapels had been churches and poor men's cottages
princes' palaces.  It is a good divine that follows his own
instructions ...</p>
```

```
<p id="mv1.2.5">If to do were as easy as to know what were
good to do, chapels had been churches and poor men's cottages
princes' palaces.  It is a good divine that follows his own
instructions ...</p>
```

The values of id attributes must be legal names with respect to the SGML declaration in force. For XML documents this means that an id value must begin with a letter (as defined in the World Wide Web Consortium's XML Recommendation) or the underscore charcter ("_"), and contain no characters other than letters, digits, hyphens, underscores, full stops, and certain combining and extension characters.[37]

For SGML documents this means that by default they must begin with a letter (from A to Z or a to z) and contain no characters other than letters, digits 0 to 9, full stop, and hyphen. By default, i.e. using the TEI-supplied SGML declaration, SGML names must be 32 or fewer characters long.

[35] More exactly, these are the attributes of the element class global, to which all elements belong; for further discussion of attribute classes and ways in which attributes may be inherited and over-ridden, see section 3.7.1 *Classes Which Share Attributes*.

[36] A dummy element class TEIform is defined in the reference section, solely for documentary purposes.

[37] The colon is also by default a valid name character; however, it is reserved for a specific purpose in XML (to indicate namespace prefixes), and is not therefore generally recommended by these Guidelines, for compatability reasons.

Furthermore, by default upper and lower case letters are not distinguished in SGML names: thus, the strings ‘a23’ and ‘A23’ are identical, and may not be used to identify two distinct elements. This may (and perhaps should) be changed in the SGML declaration.

In XML names (and thus the values of id in an XML TEI document) upper and lower case letters are distinguished, and thus ‘partTime’ and ‘parttime’ are two distinctly different names, and could (perhaps unwisely) be used to denote two different element types. This cannot be changed.

If two elements are given the same identifier, the parser will signal a syntax error. The following example, therefore, is *not* valid:

```
<p id="PAGE1"><q>What's it going to be then, eh?</q></p>
<p id="PAGE1">There was me, that is Alex, and my three droogs,
that is Pete, Georgie, and Dim, ... </p>
```

For a discussion of methods of providing unique identifiers for elements, see section 6.9.2 *Creating New Reference Systems*.

The n attribute allows identifying information (e.g. chapter numbers, etc.) to be encoded even if it would not be a legal id value. Its value may be any string of characters; typically it is a number or other similar enumerator or label. For example, the numbers given to the items of a numbered list may be recorded with the n attribute; this would make it possible to record errors in the numeration of the original, as in this list of chapters, transcribed from a faulty original in which the number 10 is used twice, and 11 is omitted:

```
<list type="ordered">
  <item n="1">About These Guidelines</item>
  <item n="2">A Gentle Introduction to SGML</item>
  <!-- ... -->
  <item n="9">Verse</item>
  <item n="10">Drama</item>
  <item n="10">Spoken Materials<!-- sic:  original has '10' twice! --> </item>
  <item n="12">Printed Dictionaries</item>
  <!-- ... -->
</list>
```

The n attribute may also be used to record non-unique names associated with elements in a text, possibly together with a unique identifier as in the following example start-tags:

```
<div type='chap' n='One' id='TXT0101'>
<div type='stanza' n='xlii'>
```

The lang attribute indicates the language, writing system, and character set associated with a given element and all its contents. If it is not specified, the value is inherited from that of the immediately enclosing element. As a rule, therefore, it is simplest to specify the base language of the text on the <TEI.2> element, and allow most elements to take the default value for lang; the language of an element then need be explicitly specified only for elements in languages other than the base language.

The following two encodings convey the same information about the language of the text, since in the first the lang attributes on the <emph> elements specify the same value as that on the parent <p> element, while in the second they inherit that value without specifying it.

```
<p lang="en"> ... Both parties deprecated war, but one of
 them would <emph lang="en">make</emph> war rather than let
 the nation survive, and the other would <emph lang="en">accept
 </emph> war rather than let it perish, and the war came.</p>
<p lang="en"> ... Both parties deprecated war, but one of
 them would <emph>make</emph> war rather than let
 the nation survive, and the other would <emph>accept</emph>
 war rather than let it perish, and the war came.</p>
```

In the following example, by contrast, the lang attribute on the <term> element must be given if we wish to record the fact that the technical terms used are Latin rather than English; no lang attribute is needed on the <q> element, by contrast, because it is in the same language as its parent. It is strongly recommended

that all language shifts in the source be explicitly identified by use of the lang attribute, as described in chapter 4 *Languages and Character Sets*.

```
<p lang="en">The constitution declares <q>that no bill of attainder
or <term lang="la">ex post facto</term> law shall be passed.</q> ... </p>
```

Formally, the lang attribute is an IDREF; a reference to the id value of a <language> element in the TEI header.[38] This means that each language used in the document should be declared in the TEI header using the <language> element defined in section 5.4.2 *Language Usage*.

The rend attribute is used to give information about the physical presentation of the text in the source. In the following example, it is used to indicate that both the emphasized word and the proper name are printed in italics:

```
<p> ... Their motives <emph rend="italics">might</emph> be
 pure and pious; but he was equally alarmed by his knowledge
 of the ambitious <name rend="italics">Bohemond</name>, and
 his ignorance of the Transalpine chiefs: ...</p>
```

If all or most <emph> and <name> elements are rendered in the text by italics, it will be more convenient to register that fact in the TEI header once and for all and specify a rend value only for any elements which deviate from the usual rendition.

The contents of the rend attribute are free text. In any given project, encoders are advised to settle on a standard vocabulary with which to describe typographic or manuscript rendition of the text, and to document their usage of that vocabulary in the <rendition> element of the TEI header.

The TEIform attribute is used to allow application programs to handle TEI-encoded documents correctly even if some or all elements have been renamed. Most users can ignore this attribute entirely; it is only relevant when the TEI DTDs are modified.[39]

The default value of TEIform for any element is the generic identifier of that element, as described in this document. The value for <p> is 'p', the value for <div1> is 'div1', etc. When elements are renamed, as described in chapter 29 *Modifying and Customizing the TEI DTD*, the declaration of TEIform is not modified. If <div1> is renamed <chapter>, for example, the default value of TEIform remains 'div1'. An application program which does not recognize the new generic identifier can check to see whether the attribute TEIform exists, and examine its value if it does to find out which TEI element, if any, is being used.

Modifications of DTDs, however, may involve more than simple renaming of elements: sometimes elements are given not just new names, but complete new definitions. In such cases, the TEIform attribute may be used to indicate the standard TEI element corresponding to the modified element. For example, if a local modification of a DTD renamed the <div1> element as <chapter> and also modified its formal declarations (e.g. to change its content model), then the TEIform attribute on the modified element should be given the default value div1, in order to indicate that the local <chapter> element is a modification of the standard TEI <div1>.

When new elements are introduced, they may be identified as specialized variants of existing TEI elements by giving them the appropriate default value for TEIform. For example, if a local element called <quatrain> were introduced, as a specialized variant of the <lg> (line group) element which must contain exactly four lines, then its declaration might give its TEIform as lg, to signify that a quatrain is a particular type of line group, thus:

```
<!ELEMENT quatrain - O  (l, l, l, l)            >
<!ATTLIST quatrain      %a.global;
          TEIform       (lg)            'lg'    >
```

The formal definition of the global attributes is as follows:

[38] Validation checks that all IDREF values exist as id values on elements somewhere in the current SGML document. It is a requirement of the TEI scheme, not of SGML or XML, that the lang attribute point to a <language> element.
[39] The TEIform attribute is based on the notion of *architectural forms* developed for HyTime (ISO 10744).

```
<!-- 3.5: Global attributes-->
<!--The global attributes are defined for every element in the TEI
tag set; individual declarations may be overridden by local declarations
for individual elements.-->
<!--If the tag sets invoked by the user define extra global
attributes (they do this in their .ent file), then they are inherited by
GLOBAL; otherwise the parameter entities referred to expand to the empty
string, as shown here.  -->
<!ENTITY % a.analysis ''>
<!ENTITY % a.linking ''>
<!ENTITY % a.terminology ''>
<!ENTITY % a.global '
      %a.terminology;
      %a.linking;
      %a.analysis;
      id ID #IMPLIED
      n CDATA #IMPLIED
      lang IDREF %INHERITED;
      rend CDATA #IMPLIED'>
<!--The TEIform attribute is also global, but is declared
individually for each element, not in a parameter entity
declaration.-->
<!-- end of 3.5-->
```

3.6 The TEI2.DTD File

All TEI-encoded documents use the same top-level DTD file, which refers to a number of other DTD files, the exact set of other files referred to depending on which base and which additional tagsets are in use. The remainder of this chapter describes in some detail the organization and function of this file and those it embeds; it is necessarily of a rather technical and specialized nature.

The main TEI DTD is always invoked by specifying the file tei2.dtd. This file:

1. takes care of certain necessary preliminaries:
    1. embeds any locally defined changes to the standard TEI parameter entities, so that local modifications can take precedence over default declarations;
    2. declares TEI-specific keywords used in other declarations and declares default values of IGNORE for all the parameter entities used to select base and additional tag sets (see section 3.8.3 *Parameter Entities for TEI Keywords*);
    3. declares parameter entities for TEI generic identifiers (by embedding the file teigis2.ent; see section 3.8.2 *Parameter Entities for Element Generic Identifiers*);
    4. declares parameter entities used to control whether the target DTD should be in SGML or XML (see section 3.8.4 *Generation of an XML DTD*).
2. declares parameter entities for element classes, content models, and global attributes (by embedding teiclas2.ent; see section 3.7.3 *The TEICLAS2.ENT File*);
3. declares the top-level elements <TEI.2> and <teiCorpus.2>;
4. embeds DTD files containing local modifications (if any), the core tag sets, the base tag set, and the additional tag sets.

3.6.1 Structure of the TEI2.DTD File

Each parameter entity associated with a tag set controls several marked sections in the main DTD file tei2.dtd. If the entity has been declared in the DTD subset with the text INCLUDE, then the marked sections it controls will be parsed; otherwise, they will be ignored. The marked sections controlled by each entity:

1. declare and refer to the entity file for the tag set, which defines its global attributes and element classes;
2. declare and refer to the DTD file for the tag set, which defines its elements and their attributes;

3. declare the parameter entity component in a form suitable for texts using that base.

The tei2.dtd file has the following structure:

```
<!-- 3.6.1: File tei2.dtd:  Main document type declaration file-->
<!--Text Encoding Initiative Consortium:
Guidelines for Electronic Text Encoding and Interchange.
Document TEI P4, 2002.
Copyright (c) 2002 TEI Consortium. Permission to copy in any form
is granted, provided this notice is included in all copies.
These materials may not be altered; modifications to these DTDs should
be performed only as specified by the Guidelines, for example in the
chapter entitled 'Modifying the TEI DTD'
These materials are subject to revision by the TEI Consortium. Current versions
are available from the Consortium website at http://www.tei-c.org-->
<!--This file first defines some useful entities, then defines the
element TEI.2 and includes files with the various specialized parts of
the document type declaration.  It also declares the top-level TEI.2
and teiCorpus.2 elements.-->
<!--I.  Preliminaries.-->
<!--Embed any local modifications to TEI entities.-->
<!--declarations from 3.6.2: Local modifications to parameter entities inserted here -->
<!--Embed entities for TEI generic identifiers.-->
<!ENTITY % TEI.elementNames PUBLIC '-//TEI P4//ENTITIES Generic
Identifiers//EN' 'teigis2.ent' >%TEI.elementNames;
<!--Define entities for TEI keywords.-->
<!--This includes defining the default for each base and additional
tag set as 'IGNORE', and initialising the tag omissibility indicator
entities depending on the value of TEI.XML-->
<!--declarations from 3.8.5: TEI Keywords inserted here -->
<!--II. Define element classes for content models, shared
attributes for element classes, and global attributes.  (This all
happens within the file TEIclas2.ent.)-->
<!ENTITY % TEI.elementClasses PUBLIC '-//TEI P4//ENTITIES TEI
ElementClasses//EN' 'teiclas2.ent' >%TEI.elementClasses;
<!--III.  Define the top-level TEI elements:  one for individual
texts, one for composites with a collective header.-->
<!--A TEI document is a text preceded by a TEI header.-->
<!ELEMENT TEI.2 %om.RO;  (teiHeader, text)>
<!ATTLIST TEI.2
      %a.global;
      TEIform CDATA 'TEI.2'  >
<!--A TEI corpus is a series of TEI.2 documents, preceded by a
corpus-level TEI header.-->
<!ELEMENT teiCorpus.2 %om.RO;  (teiHeader, TEI.2+)>
<!ATTLIST teiCorpus.2
      %a.global;
      TEIform CDATA 'teiCorpus.2'  >
<!--IV.  Embed the actual tag sets.  First embed any local
modifications and extensions.  Then embed the core tag sets, the
(single) base tag set, and the (optional) additional tag sets specified
by the user.-->
<!--declarations from 3.6.2: Embed local element declarations, etc. inserted here -->
<!--declarations from 3.6.3: Embed the core tag sets inserted here -->
<!--declarations from 3.6.4: Embed base tag set inserted here -->
<!--declarations from 3.6.5: Embed additional tag sets inserted here -->
<!-- end of 3.6.1-->
```

A TEI-conformant document *must* use the tei2.dtd file, or one derived from it in the manner described in chapter 29 *Modifying and Customizing the TEI DTD*. It must also specify which base and which additional tag sets are to be invoked, using the mechanisms described in section 3.3 *Invocation of the TEI DTD*.

3.6.2 Embedding Local Modifications

As noted above in section 3.2.4 *User-Defined Tag Sets*, local modifications to the DTD are most conveniently grouped into two files, one containing modifications to the TEI parameter entities, and

the other new or changed declarations of elements and their attributes. These files should be associated with the parameter entities TEI.extensions.ent and TEI.extensions.dtd by declarations included in the document's DTD subset.

For example, if the relevant files are called project.ent and project.dtd, then declarations like the following would be appropriate:

```
<!ENTITY % TEI.extensions.ent SYSTEM 'project.ent' >
<!ENTITY % TEI.extensions.dtd SYSTEM 'project.dtd' >
```

When an entity is declared more than once, the first declaration is binding and the others are ignored. The local modifications to parameter entities should therefore be handled before the standard parameter entities themselves are declared in tei2.dtd. The entity TEI.extensions.ent is referred to before any TEI declarations are handled, to allow the user's declarations to take priority. If the user does not provide a TEI.extensions.ent entity, the entity will be expanded to the empty string.

For example the encoder might wish to add two phrase-level elements <it> and <bd>, perhaps as synonyms for <hi rend='italics'> and <hi rend='bold'>. As described in chapter 29 *Modifying and Customizing the TEI DTD*, this involves two distinct steps: one to define the new elements, and the other to ensure that they are placed into the TEI document structure at the right place. We deal with the second first, by specifying the element class to which the new elements should be attached. To do this, the standard parameter entity x.phrase should be modified to include the two new generic identifiers. The file containing local declarations of the standard parameter entities will thus contain a declaration of the following form:

```
<!ENTITY % x.phrase 'it | bd |' >
```

The relevant fragment of the DTD is this:

```
<!-- 3.6.2: Local modifications to parameter entities-->
<!--Embed local modifications to TEI parameter entities.  Declare
entity as empty string first, in case user has no mods and has not
declared it.-->
<!ENTITY % TEI.extensions.ent '' >%TEI.extensions.ent;
<!-- end of 3.6.2-->
```

The second type of modification needed is most conveniently performed after all the standard TEI parameter entities have been declared; this allows the element declarations provided by the user to make use of the parameter entities which define standard TEI content models and attribute definitions. To facilitate this, the parameter entity TEI.extensions.dtd is used to embed local element declarations *before* any of the TEI tag sets are embedded by the file tei2.dtd, but *after* all the TEI element classes and other parameter entities have been declared.

The task of declaring the non-standard <it> and <bd> elements is thus simplified: they can, for example, use the same parameter entities as the <hi> element. A suitable local DTD-modifications file might look like the following (note that the standard parameter-entity reference for phrase sequence is used):

```
<!ELEMENT it           (%phrase.seq;)                      >
<!ATTLIST it
          id               ID            #IMPLIED
          lang             IDREF         %INHERITED;
          n                CDATA         #IMPLIED
          rend             CDATA         #FIXED 'italics'
          TEIform          CDATA         "hi"              >
<!ELEMENT bd           (%phrase.seq;)                      >
<!ATTLIST bd
          id               ID            #IMPLIED
          lang             IDREF         %INHERITED;
          n                CDATA         #IMPLIED
          rend             CDATA         #FIXED 'boldface'
          TEIform          CDATA         "hi"              >
```

For further examples of local modifications to both parameter entities and element declarations, see chapter 29 *Modifying and Customizing the TEI DTD*.

The relevant fragment of the DTD is this:

```
<!-- 3.6.2: Embed local element declarations, etc.-->
<!--Embedding local modifications here allows user modifications
to use all the standard TEI element classes and parameter entities.-->
<!ENTITY % TEI.extensions.dtd '' >%TEI.extensions.dtd;
<!-- end of 3.6.2-->
```

3.6.3 Embedding the Core Tag Sets

The core tag sets are embedded by the file tei2.dtd using the parameter entities TEI.header and TEI.core. The relevant fragment of the DTD is this:

```
<!-- 3.6.3: Embed the core tag sets-->
<!--These occur in all documents and are therefore defined
unconditionally.-->
<!ENTITY % TEI.header.dtd PUBLIC '-//TEI P4//ELEMENTS TEI Header//EN'
'teihdr2.dtd' >%TEI.header.dtd;

<!ENTITY % TEI.core.dtd PUBLIC '-//TEI P4//ELEMENTS Core Elements//EN'
'teicore2.dtd' >%TEI.core.dtd;
<!-- end of 3.6.3-->
```

The default text structure tags, which are also documented as part of the core, are embedded by the base tag set, unless the base defines its own text structure tags; see the chapters on the individual bases.

3.6.4 Embedding the Base Tag Set

The tei2.dtd file embeds the appropriate files for the base tag set previously selected by means of the parameter entities described in section 3.2 *Core, Base, and Additional Tag Sets*. A parameter entity for the file containing the relevant DTD fragment is declared and referred to inside a conditional marked section controlled by the appropriate parameter entity. The relevant fragment of tei2.dtd is this:

```
<!-- 3.6.4: Embed base tag set-->
<!--A different base will be embedded, depending on which
parameter entity has been declared by the user
with the value 'INCLUDE'.-->
<![%TEI.prose;[
<!ENTITY % TEI.prose.dtd PUBLIC '-//TEI P4//ELEMENTS Base Element Set for
Prose//EN' 'teipros2.dtd' >
%TEI.prose.dtd;
]]>
<![%TEI.verse;[
<!ENTITY % TEI.verse.dtd PUBLIC '-//TEI P4//ELEMENTS Base Element Set for
Verse//EN' 'teivers2.dtd' >
%TEI.verse.dtd;
]]>
<![%TEI.drama;[
<!ENTITY % TEI.drama.dtd PUBLIC '-//TEI P4//ELEMENTS Base Element Set for
Drama 2001-12//EN' 'teidram2.dtd' >
%TEI.drama.dtd;
]]>
<![%TEI.spoken;[
<!ENTITY % TEI.spoken.dtd PUBLIC '-//TEI P4//ELEMENTS Base Element Set for
Transcriptions of Speech//EN' 'teispok2.dtd' >
%TEI.spoken.dtd;
]]>
<![%TEI.dictionaries;[
<!ENTITY % TEI.dictionaries.dtd PUBLIC '-//TEI P4//ELEMENTS Base Element
Set for Print Dictionaries//EN' 'teidict2.dtd' >
%TEI.dictionaries.dtd;
]]>
<![%TEI.terminology;[
<!ENTITY % TEI.terminology.dtd PUBLIC '-//TEI P4//ELEMENTS Base Element
Set for Terminological Data//EN' 'teiterm2.dtd' >
%TEI.terminology.dtd;
]]>
<![%TEI.general;[
<!ENTITY % TEI.general.dtd PUBLIC '-//TEI P4//ELEMENTS General Base
Element Set//EN' 'teigen2.dtd' >
```

```
%TEI.general.dtd;
]]>
<![%TEI.mixed;[
<!ENTITY % TEI.mixed.dtd PUBLIC '-//TEI P4//ELEMENTS Base Element Set for
Mixed Text Types//EN' 'teimix2.dtd' >
%TEI.mixed.dtd;
]]>
<!-- end of 3.6.4-->
```

3.6.5 Embedding the Additional Tag Sets

The tei2.dtd file embeds the appropriate files for any additional base tag set previously enabled by means of the parameter entities described in section 3.2 *Core, Base, and Additional Tag Sets*. A parameter entity for the file containing the relevant DTD fragment is declared and referred to, inside a conditional marked section controlled by the appropriate parameter entity. The relevant fragment of tei2.dtd is this:

```
<!-- 3.6.5: Embed additional tag sets-->
<!--These entities are declared and embedded only when the user
has overridden the default declaration of IGNORE for a specific
additional tag set.-->
<![%TEI.linking;[
<!ENTITY % TEI.linking.dtd PUBLIC '-//TEI P4//ELEMENTS Additional
Element Set for Linking, Segmentation, and Alignment//EN'
'teilink2.dtd' >
%TEI.linking.dtd;
]]>
<![%TEI.analysis;[
<!ENTITY % TEI.analysis.dtd PUBLIC '-//TEI P4//ELEMENTS Additional
Element Set for Simple Analysis//EN' 'teiana2.dtd' >
%TEI.analysis.dtd;
]]>
<![%TEI.fs;[
<!ENTITY % TEI.fs.dtd PUBLIC '-//TEI P4//DTD Auxiliary Document Type:
 Feature System Declaration//EN' 'teifs2.dtd' >
%TEI.fs.dtd;
]]>
<![%TEI.certainty;[
<!ENTITY % TEI.certainty.dtd PUBLIC '-//TEI P4//ELEMENTS Additional
Element Set for Certainty and Responsibility//EN' 'teicert2.dtd' >
%TEI.certainty.dtd;
]]>
<![%TEI.transcr;[
<!ENTITY % TEI.transcr.dtd PUBLIC '-//TEI P4//ELEMENTS Additional
Element Set for Transcription of Primary Sources//EN' 'teitran2.dtd' >
%TEI.transcr.dtd;
]]>
<![%TEI.textcrit;[
<!ENTITY % TEI.textcrit.dtd PUBLIC '-//TEI P4//ELEMENTS Additional
Element Set for Text-Critical Apparatus//EN' 'teitc2.dtd' >
%TEI.textcrit.dtd;
]]>
<![%TEI.names.dates;[
<!ENTITY % TEI.names.dates.dtd PUBLIC '-//TEI P4//ELEMENTS Additional
Element Set for Names and Dates//EN' 'teind2.dtd' >
%TEI.names.dates.dtd;
]]>
<![%TEI.nets;[
<!ENTITY % TEI.nets.dtd PUBLIC '-//TEI P4//ELEMENTS Additional Element
Set for Graphs, Networks, and Trees//EN' 'teinet2.dtd' >
%TEI.nets.dtd;
]]>
<![%TEI.figures;[
<!ENTITY % TEI.figures.dtd PUBLIC '-//TEI P4//ELEMENTS Additional
Element Set for Tables, Formulae, and Graphics//EN' 'teifig2.dtd' >
%TEI.figures.dtd;
]]>
```

```
<![%TEI.corpus;[
<!ENTITY % TEI.corpus.dtd PUBLIC '-//TEI P4//ELEMENTS Additional Element
Set for Language Corpora//EN' 'teicorp2.dtd' >
%TEI.corpus.dtd;
]]>
<!-- end of 3.6.5-->
```

3.7 Element Classes

The TEI DTD contains over four hundred element types. To aid comprehension, modularity and modification, the majority of these elements are formally classified in some way. This section describes the various *element classes* recognized in the TEI DTD. Element classes are used to express two distinct kinds of commonality among elements. The elements of a class may share some set of attributes, or they may appear in the same locations in the content models of the TEI DTDs, or both. A class is known as an *a-class* if its members share attributes, and as an *m-class* if its members appear at the same locations in the content models of other TEI elements. An element is said to *inherit* attributes, or the ability to appear at a given point in a document, from any classes of which it is a member. Classes may have subclasses and superclasses, and the characteristics of a superclass are inherited by all members of its subclasses.

Both types of element classes are represented in the TEI DTDs by parameter entities. For other uses of parameter entities in the TEI DTDs, see section 3.8 *Other Parameter Entities in TEI DTDs*.

This section describes the major element classes of each type together with the formal declarations for their parameter entities, which are contained in the file teiclas2.ent. All element classes are documented in the alphabetical reference section in Part VII.

3.7.1 Classes Which Share Attributes

An attribute class (a-class) groups together elements which share some set of common attributes. For example, the members of the class names are all elements which contain proper nouns: e.g. <name>, <placeName>, or <persName>. All of these elements use the same attributes (key and reg) to record information about the referent or the regularized form of the proper nouns. Similarly, the members of the pointer class share a set of attributes useful for managing cross-reference links and other pointers.[40]

The attributes shared by the members of an a-class are defined in a parameter entity; member elements inherit the attributes by referring to the parameter entity within their attribute-list declaration (examples below). This practice helps ensure that if the attribute definitions for the class change, all members of the class will automatically inherit the new definitions. Parameter entities used for this purpose form their names by taking the name of the class they define and prefixing the string 'a.'; we refer to these entities as *a-dot entities.*

For example, the declaration for the names class includes attribute definitions for its two attributes reg and key:

```
<!-- 3.7.1: Sample class declaration-->
<!ENTITY % a.names '
      key CDATA #IMPLIED
      reg CDATA #IMPLIED'>
<!-- end of 3.7.1-->
```

Members of the class typically inherit these definitions by referring to a.names:

```
<!-- 3.7.1: Sample element declarations-->
<!--The members of a class refer to its a-dot entity in their
attribute-list declaration, as shown here.-->
<!ELEMENT name %om.RR;  %phrase.seq;>
<!ATTLIST name
      %a.global;
      %a.names;
      type CDATA #IMPLIED
      TEIform CDATA 'name'  >
<!-- end of 3.7.1-->
```

[40] Because the details of their pointing mechanism differ, the members of the pointer class do not, however, share their pointing attributes.

Subclasses of a-classes inherit the attributes of their superclass similarly, by referring to the a-dot entity of the superclass in defining their own a-dot entity. For example, the class xPointer is a subclass of the class pointer, as shown implicitly by the declaration of its a-dot entity:

```
<!-- 3.7.1: Sample subclass declaration, with inheritance from superclass-->
<!ENTITY % a.xPointer '
      %a.pointer;
      doc ENTITY #IMPLIED
      from %extPtr; "ROOT"
      to %extPtr; "DITTO"'>
<!-- end of 3.7.1-->
```

(For an explanation of the parameter entity extptr used in the above example, see section 3.8.3 *Parameter Entities for TEI Keywords*.)

The a-classes declared in the core tag sets of these Guidelines are:

declaring elements which have a decls attribute for specifying which declarations in the header apply to the element, as described in section 23.3 *Associating Contextual Information with a Text*

declarable header elements containing declarations, which can be pointed at by the decls attribute, as described in section 23.3 *Associating Contextual Information with a Text*

divn structural elements which behave in the same way as divisions, as described in section 7.1 *Divisions of the Body*

enjamb elements which carry the enjamb attribute for indicating metrical enjambement

interpret elements which contain overtly interpretive or extra-textual analysis or commentary on a text or some portion of it

metrical elements which carry metrical information (metrical pattern, realization of the pattern, rhyme)

names elements which contain proper nouns and share attributes for identifying their referents and regularizing their spelling (section 6.4.1 *Referring Strings*)

personPart elements which contain personal names or parts of them

placePart elements which contain place names or parts of them

pointer elements which point from one location in the document to another (section 6.6 *Simple Links and Cross References*)

seg elements for the systematic or arbitrary segmentation of the text

temporalExpr elements which contain temporal expressions

timed elements (in the base tag set for spoken texts) which have a duration in time expressible with the attributes, as described in section 11.2.5 *Temporal Information*

typed elements which carry an additional semantic or functional classification

xPointer elements which point from one location in the document to other locations within or outside the current document (section 14.2 *Extended Pointers*)

All elements are considered members of the class global and thus include a reference to a.global; in their attribute definition list declaration. Some tag sets add specialized attributes to the set of global attributes; these additions are declared in the 'ent' file of each tag set, using the following entity names. If the tag set does not define new global attributes, no entity of this type is declared.

a.analysis additional global attributes for the analysis tag set

a.linking additional global attributes for the linking tag set

a.terminology additional global attributes for the terminology base

These entities are included in the teiclas2.ent file indirectly, when the entity-declaration files of each tag set are embedded, as shown below in section 3.7.6 *Elements Marked for Text Type*. For purposes of documentation, these attributes are treated as if inherited by the class global from superclasses called terminology, etc., and are documented under the class name.

```
<!-- 3.7.1: Attribute classes-->
<!ENTITY % a.declaring '
      decls IDREFS #IMPLIED'>
<!ENTITY % a.declarable '
      default ( YES | NO ) "NO"'>
<!ENTITY % a.typed '
      type CDATA #IMPLIED
      subtype CDATA #IMPLIED'>
<!ENTITY % a.enjamb ''>
<!ENTITY % a.interpret '
      resp CDATA %INHERITED;
      type CDATA %INHERITED;
      inst IDREFS #IMPLIED'>
<!ENTITY % a.metrical ''>
<!ENTITY % a.divn '
      %a.metrical;
      type CDATA #IMPLIED
      org (composite | uniform) "uniform"
      sample (initial | medial | final | unknown | complete) "complete"
      part (Y | N | I | M | F) "N"'>
<!ENTITY % a.names '
      key CDATA #IMPLIED
      reg CDATA #IMPLIED'>
<!ENTITY % a.personPart ''>
<!ENTITY % a.placePart ''>
<!ENTITY % a.pointer '
      type CDATA #IMPLIED
      resp CDATA #IMPLIED
      crdate CDATA #IMPLIED
      targType CDATA #IMPLIED
      targOrder (Y | N | U) "U"
      evaluate ( all | one | none ) #IMPLIED'>
<!ENTITY % a.seg '
      %a.metrical;
      type CDATA #IMPLIED
      function CDATA #IMPLIED
      part (Y | N | I | M | F) "N"'>
<!ENTITY % a.temporalExpr ''>
<!ENTITY % a.timed '
      start IDREF #IMPLIED
      end IDREF #IMPLIED
      dur CDATA #IMPLIED'>
<!ENTITY % a.xPointer '
      %a.pointer;
      doc ENTITY #IMPLIED
      from %extPtr; "ROOT"
      to %extPtr; "DITTO"'>
<!-- end of 3.7.1-->
```

3.7.2 Classes Used in Content Models

When the members of a class are structurally similar and can appear at the same kinds of structural locations in the document, they are grouped together into an m-class (or 'model-class'). M-classes are implemented by defining a parameter entity for use in the formal declaration of element content models. The parameter entity takes the name of the class it defines, and prefixes the string 'm.', which can be interpreted as *model* or as *members*. The replacement text of the entity is a list of the members of the class, separated by '|', the content model symbol for alternation.

For each class an additional entity is defined, which also takes the name of the class, this time prefixed by the string 'x.' (for extension); the default value of these *x-dot entities* is always an empty string. A reference to the corresponding x-dot entity is always included within the replacement string for each m-dot entity. This enables an encoder to add new members to a class simply by declaring a new value for an x-dot entity.

For example, the class bibl has the three members <bibl>, <biblFull>, and <biblStruct>. Its content-model entity is defined thus:

```
<!ENTITY % x.bibl '' >
<!ENTITY % m.bibl '%x.bibl; bibl | biblFull | biblStruct' >
```

With the default value of the x-dot entity, this is the same as defining m.bibl with the replacement text `bibl | biblFull | biblStruct`. If an encoder wishes to add a new bibliographic element called <my.bib>, it can be added to the bibl class by redefining the x-dot entity thus:

```
<!ENTITY % x.bibl 'my.bib |' >
```

This changes the replacement text of m.bibl from its default value to `my.bib | bibl | biblFull | biblStruct`. If more than one element is to be added to a class, the x-dot entity for the class should be redefined as a list of the new generic identifiers, each one *(including the last)* followed by a vertical bar. The same effect could be achieved simply by redefining the whole of the new m.bibl entity directly, but the x-dot method requires no repetition of the already existing members of the class and thus minimizes the chance of error.

Like a-classes, m-classes may have subclasses or superclasses. Just as elements inherit from a class the ability to appear in certain locations of a document (wherever the class can appear), so all members of a subclass inherit the ability to appear wherever any superclass can appear. Superclasses transmit their location characteristics to their subclasses by referring, in declaring their m-dot entity, to the m-dot entities of the subclasses.

For example, the class phrase includes the classes data, edit, hqphrase, loc, and seg as members, as can be seen in the declaration for its m-dot entity:

```
<!-- 3.7.2: Sample class declaration with inclusion of subclasses-->
<!ENTITY % x.phrase "" >
<!ENTITY % m.phrase "%x.phrase; %m.data; | %m.edit; | %m.formPointers; |
%n.formula; | %n.fw; | %n.handShift; | %m.hqphrase; | %m.loc; |
%m.phrase.verse; | %m.seg; | %m.sgmlKeywords;">
<!-- end of 3.7.2-->
```

When the entity m.phrase is referred to in content models, all members of all subclasses are included in the model.

3.7.3 The TEICLAS2.ENT File

The most important element classes used in TEI content models are declared in the DTD file teiclas2.ent, which is the default replacement text for the entity TEI.elementClasses and is embedded by the tei2.dtd file. These element classes are described, and their declarations reproduced, in the following sections.

The class system is structured around the following threefold division of elements:

chunks elements such as paragraphs and other paragraph-level elements, which can appear directly within texts or within text subdivisions (i.e. elements), but not within other chunks

phrase-level elements elements such as highlighted phrases, book titles, or editorial corrections which can occur only within chunks (paragraphs or paragraph-level elements), but not between them (and thus cannot appear directly within a)[41]

inter-level elements elements such as lists, notes, quotations, etc. which can appear either between chunks (as children of a) or within them

Together the two sets of *chunks* and *inter-level elements* make up the set of:

text components elements which can appear directly within texts or text divisions; also called simply *components* or 'component-level elements'

[41] Note that in this context, *phrase* means any string of characters, and can apply to individual words, parts of words, and groups of words indifferently; it does not refer only to linguistically motivated phrasal units. This may cause confusion for readers accustomed to applying the word in a more restrictive sense.

In general, the body of any text comprises a series of components, optionally grouped into elements.

Some elements belong to none of these classes; these include high-level structural elements like <TEI.2> and <group> as well as some specialized elements which appear only within particular structures (like <analytic>, <monographic>, and <series>). The majority of elements found in normal running text, however, are assigned by the TEI DTDs to one or the other of these classes.

Some *component* elements (e.g. <p> or <note>) are common to all base tag sets, while others are unique to individual tag sets. This distinction is reflected in the parameter entity declarations, as shown below.

The teiclas2.ent file has the following overall structure:

```
<!-- 3.7.3: Element classes for TEI DTDs-->
<!--Text Encoding Initiative Consortium:
Guidelines for Electronic Text Encoding and Interchange.
Document TEI P4, 2002.
Copyright (c) 2002 TEI Consortium. Permission to copy in any form
is granted, provided this notice is included in all copies.
These materials may not be altered; modifications to these DTDs should
be performed only as specified by the Guidelines, for example in the
chapter entitled 'Modifying the TEI DTD'
These materials are subject to revision by the TEI Consortium. Current versions
are available from the Consortium website at http://www.tei-c.org-->
<!--First, we declare the 'low-level' core classes:
these are classes of semantically and structurally similar elements
declared as part of the core tag set, e.g. the classes 'data' or
'edit'.-->
<!--declarations from 3.7.4: Low-level classes inserted here -->
<!--declarations from 3.7.9: Misc. Element Class Models inserted here -->
<!--Next, we declare the 'high-level' classes: these group
together all phrase-level elements, all inter-level elements, and all
chunk-level elements in the core, and identify the 'common' component
elements (chunks and inter-level elements), as opposed to the
tagset-specific components.-->
<!--declarations from 3.7.5: Common high-level classes inserted here -->
<!--Next, we embed the portions of each base and additional tag
set which declare relevant parameter entities. Only those files are
embedded which have been selected by the user in the DTD subset. These
files will declare parameter entities for their component-level
elements, as well as for any global attributes they define.-->
<!--declarations from 3.7.6: Embedding tag-set-specific entity definitions inserted here -->
<!--We can now declare the standard content models; one of these
varies with the base selected.-->
<!--declarations from 3.7.7: Standard Content Models inserted here -->
<!--Finally, we declare the attribute classes, including
the global attributes.-->
<!--declarations from 3.7.1: Attribute classes inserted here -->
<!--declarations from 3.5: Global attributes inserted here -->
<!-- end of 3.7.3-->
```

3.7.4 Low-Level Element Classes

The following low-level classes group together sets of semantically or structurally similar elements. These classes may include both elements in the core and elements declared in particular tag sets; a reference is given at least to the relevant section on the core tags.

The following are phrase-level element classes:

hqphrase elements for highlighted phrases or material marked by quotation marks, including those defined in section 6.3 *Highlighting and Quotation*

data elements for recording information about the referents of a text, including those defined in section 6.4 *Names, Numbers, Dates, Abbreviations, and Addresses*

date elements for recording dates, including those defined in section 6.4.4 *Dates and Times*

edit elements for recording simple editorial interventions in a text, including those defined in section 6.5 *Simple Editorial Changes*

loc elements for recording location information in a text, including those defined in section 6.9 *Reference Systems*

seg elements for marking arbitrary segments at the level of individual characters or phrases, including those documented in section 14.3 *Blocks, Segments and Anchors* and 15.1 *Linguistic Segment Categories*

sgmlKeywords elements for marking generic identifiers, attribute names, tags, and sample attribute values, when they occur in the text (used in tag set documentation, for which see chapter 27 *Tag Set Documentation*)

versePhrases phrase-level elements specific to verse, documented in section 9.3 *Components of the Verse Line*

formPointers elements for referring, within a dictionary entry, to the orthographic form or pronunciation of the headword, documented in section 12.4 *Headword and Pronunciation References*

The following are inter-level element classes:

hqinter elements for highlighted phrases or material marked by quotation marks, including those defined in section 6.3 *Highlighting and Quotation*

bibl elements for bibliographic citations; see section 6.10 *Bibliographic Citations and References*

lists elements for lists; see section 6.7 *Lists*

notes general-purpose annotation elements; see section 6.8 *Notes, Annotation, and Indexing*

stageDirection elements for specialized stage-direction elements documented in section 10.2.3 *Stage Directions*

The following classes of elements may appear anywhere within the `<text>` element:

metadata elements which convey non-textual information about the text (meta-information, as it were)

refsys milestone elements used in reference systems

editincl elements marking arbitrary spans of text which has been added, deleted, or omitted from a transcription

These three classes together make up the Incl class, comprising elements which may appear anywhere within a `<text>` element. In earlier versions of the Guidelines, this was implemented as an SGML inclusion exception on the `<text>` element. In the current version, members of this class are explicitly added to relevant content models only.

The entity declarations for these classes are these:

```
<!-- 3.7.4: Low-level classes-->
<!--Most of these elements are in the core tag set, but
some may be from other tag sets.-->
<!--Phrase-level classes-->
<!ENTITY % x.hqphrase "" >
<!ENTITY % m.hqphrase "%x.hqphrase; %n.distinct; | %n.emph; | %n.foreign;
| %n.gloss; | %n.hi; | %n.mentioned; | %n.soCalled; | %n.term; |
%n.title;">
<!ENTITY % x.date "" >
<!ENTITY % m.date "%x.date; %n.date; | %n.dateRange; | %n.dateStruct;">
<!ENTITY % x.data "" >
<!ENTITY % m.data "%x.data; %n.abbr; | %n.address; | %n.date; |
%n.dateRange; | %n.dateStruct; | %n.expan; | %n.geogName; | %n.lang; |
%n.measure; | %n.name; | %n.num; | %n.orgName; | %n.persName; | %n.placeName; |
%n.rs; | %n.time; | %n.timeRange; | %n.timeStruct;">
<!ENTITY % x.edit "" >
<!ENTITY % m.edit "%x.edit; %n.add; | %n.app; | %n.corr; | %n.damage; |
%n.del; | %n.orig; | %n.reg; | %n.restore; | %n.sic; | %n.space; |
```

```
%n.supplied; | %n.unclear;">
<!ENTITY % x.loc "" >
<!ENTITY % m.loc "%x.loc; %n.ptr; | %n.ref; | %n.xptr; | %n.xref;">
<!ENTITY % x.seg "" >
<!ENTITY % m.seg "%x.seg; %n.c; | %n.cl; | %n.m; | %n.phr; | %n.s; | %n.seg; |
%n.w;">
<!ENTITY % x.sgmlKeywords "" >
<!ENTITY % m.sgmlKeywords "%x.sgmlKeywords; %n.att; | %n.gi; | %n.tag; |
%n.val;">
<!ENTITY % x.phrase.verse "" >
<!ENTITY % m.phrase.verse "%x.phrase.verse; %n.caesura;">
<!ENTITY % x.formPointers "" >
<!ENTITY % m.formPointers "%x.formPointers; %n.oRef; | %n.oVar; |
%n.pRef; | %n.pVar;">
<!ENTITY % x.metadata "" >
<!ENTITY % m.metadata "%x.metadata; %n.alt; | %n.altGrp; | %n.certainty; |
%n.fLib; | %n.fs; | %n.fsLib; | %n.fvLib; | %n.index; | %n.interp; |
%n.interpGrp; | %n.join; | %n.joinGrp; | %n.link; | %n.linkGrp; | %n.respons; |
%n.span; | %n.spanGrp; | %n.timeline;">
<!ENTITY % x.refsys "" >
<!ENTITY % m.refsys "%x.refsys; %n.cb; | %n.lb; | %n.milestone; | %n.pb;">
<!ENTITY % x.editIncl "" >
<!ENTITY % m.editIncl "%x.editIncl; %n.addSpan; | %n.delSpan; | %n.gap;">
<!ENTITY % x.Incl "" >
<!ENTITY % m.Incl "%x.Incl; %n.anchor; | %m.editIncl; | %m.metadata; |
%m.refsys;">
<!--Inter-level classes-->
<!ENTITY % x.hqinter "" >
<!ENTITY % m.hqinter "%x.hqinter; %n.cit; | %n.q; | %n.quote;">
<!ENTITY % x.bibl "" >
<!ENTITY % m.bibl "%x.bibl; %n.bibl; | %n.biblFull; | %n.biblStruct;">
<!ENTITY % x.lists "" >
<!ENTITY % m.lists "%x.lists; %n.label; | %n.list; | %n.listBibl;">
<!ENTITY % x.notes "" >
<!ENTITY % m.notes "%x.notes; %n.note; | %n.witDetail;">
<!ENTITY % x.stageDirection "" >
<!ENTITY % m.stageDirection "%x.stageDirection; %n.camera; |
%n.caption; | %n.move; | %n.sound; | %n.tech; | %n.view;">
<!-- end of 3.7.4-->
```

3.7.5 High-Level Element Classes

The following element classes are used to implement the threefold structural distinction among phrases, chunks, and intermediate elements discussed above in section 3.7.3 *The TEICLAS2.ENT File*. In this terminology, *chunk*s (or *chunk elements*) are elements which can occur only in chunk-level sequences (e.g. between but not within paragraphs); *inter-level elements* can occur either within chunks (at phrase-level) or between chunks (e.g. at paragraph-level), and *phrase-level elements* can occur only at phrase level, within chunks (e.g. within but not between paragraphs).

The element class common includes all component-level (chunk- and inter-level) elements common to more than one base. It is used in implementing the combined bases described in section 3.4 *Combining TEI Base Tag Sets*.

The relevant portion of the DTD looks like this:

```
<!-- 3.7.5: Common high-level classes-->
<!--These are the three fundamental element classes.-->
<!ENTITY % x.phrase "" >
<!ENTITY % m.phrase "%x.phrase; %m.data; | %m.edit; | %m.formPointers; |
%n.formula; | %n.fw; | %n.handShift; | %m.hqphrase; | %m.loc; |
%m.phrase.verse; | %m.seg; | %m.sgmlKeywords;">
<!ENTITY % x.inter "" >
<!ENTITY % m.inter "%x.inter; %m.bibl; | %n.castList; | %n.figure; |
%m.hqinter; | %m.lists; | %m.notes; | %n.stage; | %m.stageDirection; |
%n.table; | %n.text;">
<!ENTITY % x.chunk "" >
```

```
<!ENTITY % m.chunk "%x.chunk; %n.ab; | %n.eTree; | %n.graph; | %n.l; | %n.lg;
| %n.p; | %n.sp; | %n.tree; | %n.witList;">
<!--This class isolates all the common
component-level elements.-->
<!ENTITY % x.common "" >
<!ENTITY % m.common "%x.common; %m.bibl; | %m.chunk; | %m.hqinter; |
%m.lists; | %m.notes; | %n.stage;">
<!-- end of 3.7.5-->
```

3.7.6 Elements Marked for Text Type

The following element classes are used to group together component-level elements which are allowed only in texts of a particular type (i.e. texts using a specific base).

comp.verse elements unique to verse

comp.drama elements unique to drama

comp.spoken elements unique to spoken texts

comp.dictionaries elements unique to dictionaries

comp.terminology elements unique to terminological data

Declarations for these base-specific element classes are included in the *entity file* of each base, which is in turn embedded by the teiclas2.dtd file in the DTD fragment shown below. If the tag set defines additions to the set of global attributes, or declares a class of component-level elements unique to the tag set, then it has an entity file which is embedded here; otherwise not.

```
<!-- 3.7.6: Embedding tag-set-specific entity definitions-->
<![%TEI.verse;[
<!ENTITY % TEI.verse.ent PUBLIC '-//TEI P4//ENTITIES Element Classes for
Verse//EN' 'teivers2.ent' >
%TEI.verse.ent;
]]>
<![%TEI.drama;[
<!ENTITY % TEI.drama.ent PUBLIC '-//TEI P4//ENTITIES Element Classes for
Drama//EN' 'teidram2.ent' >
%TEI.drama.ent;
]]>
<![%TEI.spoken;[
<!ENTITY % TEI.spoken.ent PUBLIC '-//TEI P4//ENTITIES Element Classes for
Transcriptions of Speech//EN' 'teispok2.ent' >
%TEI.spoken.ent;
]]>
<![%TEI.dictionaries;[
<!ENTITY % TEI.dictionaries.ent PUBLIC '-//TEI P4//ENTITIES Element
Classes for Print Dictionaries//EN' 'teidict2.ent' >
%TEI.dictionaries.ent;
]]>
<![%TEI.terminology;[
<!ENTITY % TEI.terminology.ent PUBLIC '-//TEI P4//ENTITIES Element
Classes for Terminological Data//EN' 'teiterm2.ent' >
%TEI.terminology.ent;
]]>
<![%TEI.linking;[
<!ENTITY % TEI.linking.ent PUBLIC '-//TEI P4//ENTITIES Element Classes
for Linking, Segmentation, and Alignment//EN' 'teilink2.ent' >
%TEI.linking.ent;
]]>
<![%TEI.analysis;[
<!ENTITY % TEI.analysis.ent PUBLIC '-//TEI P4//ENTITIES Element Classes
for Simple Analysis//EN' 'teiana2.ent' >
%TEI.analysis.ent;
]]>
<![%TEI.transcr;[
<!ENTITY % TEI.transcr.ent PUBLIC '-//TEI P4//ENTITIES Element Classes
for Transcription of Primary Sources//EN' 'teitran2.ent' >
%TEI.transcr.ent;
```

```
]]>
<![%TEI.textcrit;[
<!ENTITY % TEI.textcrit.ent PUBLIC '-//TEI P4//ENTITIES Element Classes
for Critical Apparatus//EN' 'teitc2.ent' >
%TEI.textcrit.ent;
]]>
<![%TEI.names.dates;[
<!ENTITY % TEI.names.dates.ent PUBLIC '-//TEI P4//ENTITIES Element
Classes for Names and Dates//EN' 'teind2.ent' >
%TEI.names.dates.ent;
]]>
<![%TEI.figures;[
<!ENTITY % TEI.figures.ent PUBLIC '-//TEI P4//ENTITIES Formulae
Notations and Contents//EN' 'teifig2.ent' >
%TEI.figures.ent;
]]>
<!-- end of 3.7.6-->
```

3.7.7 Standard Content Models

As far as possible, the TEI DTDs use the following set of frequently-encountered content models to help achieve consistency among different elements.

phrase a single sequence of character data or single phrase-level element

phrase.seq sequence of character data and phrase-level elements

component a single chunk- or inter-level element

component.seq sequence of chunk- and inter-level elements; this is the usual content of a `<div>` element

paraContent sequence of character data, phrase-level elements, and inter-level elements; this is the usual content of chunks (including, most prominently, paragraphs)

specialPara specialized content model, allowing *either* a sequence of chunks *or* the same content as paraContent; this is used for elements such as notes and list items, which can behave either as chunk-level elements, or else as containers for groups of component-level elements.

The relevant portion of the DTD looks like this:

```
<!-- 3.7.7: Standard Content Models-->
<!--Here we declare the parameter entities phrase, phrase.seq,
component, component.seq, paraContent, and specialPara, for use in the
content models of element declarations.  The entities phrase and
phrase.seq are the same in all bases.  They may include elements
specific to single tag sets; if the tag set is not selected, these
elements are undefined and have no effect.-->
<!ENTITY % phrase '#PCDATA | %m.phrase; | %m.Incl;' >
<!ENTITY % phrase.seq '(%phrase;)*'  >
<!--The entity component varies with the base.  The
versions for the combined bases are declared first (so as to
take precedence over the declarations in the individual
bases).-->
<!--declarations from 3.7.8: Definition of components for combined bases inserted here -->
<![%TEI.prose;[
<!ENTITY % component '(%m.common;)' >
<!ENTITY % TEI.singleBase 'INCLUDE' >]]>
<![%TEI.verse;[
<!ENTITY % component '(%m.common; | %m.comp.verse;)' >
<!ENTITY % TEI.singleBase 'INCLUDE' >]]>
<![%TEI.drama;[
<!ENTITY % component '(%m.common; | %m.comp.drama;)' >
<!ENTITY % TEI.singleBase 'INCLUDE' >]]>
<![%TEI.spoken;[
<!ENTITY % component '(%m.common; | %m.comp.spoken;)' >
<!ENTITY % TEI.singleBase 'INCLUDE' >]]>
<![%TEI.dictionaries;[
<!ENTITY % component '(%m.common; | %m.comp.dictionaries;)' >
```

```
<!ENTITY % TEI.singleBase 'INCLUDE' >]]>
<![%TEI.terminology;[
<!ENTITY % component '(%m.common; | %m.comp.terminology;)' >
<!ENTITY % TEI.singleBase 'INCLUDE' >]]>
<!--Default declaration.-->
<!ENTITY % component '(%m.common;)' >
<!ENTITY % TEI.singleBase 'INCLUDE' >
<!--The entity component.seq is always a starred sequence of
component elements.  Its definition does not vary with the base (unless
we are using the general base, in which case it has already been defined
above), but the meaning of the definition does.-->
<!ENTITY % component.seq '((%component;), (%m.Incl;)*)*' >
<!--The following entities do not vary with the base.-->
<!ENTITY % paraContent '(#PCDATA | %m.phrase; | %m.inter; | %m.Incl;)*' >
<!ENTITY % specialPara '(#PCDATA | %m.phrase; | %m.inter; | %m.chunk; |
%m.Incl;)*' >
<!-- end of 3.7.7-->
```

3.7.8 Components in Mixed and General Bases

When the mixed or general base is in use, the definitions of the entities component and component.seq are rather more complex. The relevant portion of the DTD is this:

```
<!-- 3.7.8: Definition of components for combined bases-->
<!--Default declarations for the 'mix.' entities used for mixed
and general bases.-->
<!ENTITY % mix.verse '' >
<!ENTITY % mix.drama '' >
<!ENTITY % mix.spoken '' >
<!ENTITY % mix.dictionaries '' >
<!ENTITY % mix.terminology '' >
<![%TEI.mixed;[
<!ENTITY % TEI.singleBase 'IGNORE' >
<!--The mixed base allows components from any base; it does so by
defining 'component' as including both common components and those
specific to one tag set.-->
<!ENTITY % component '(%m.common; %mix.verse; %mix.drama; %mix.spoken;
%mix.dictionaries; %mix.terminology;)' >]]>
<![%TEI.general;[
<!--The general base uses the same definition of component as the
mixed base.-->
<!ENTITY % TEI.singleBase 'IGNORE' >
<!ENTITY % component '(%m.common; %mix.verse; %mix.drama; %mix.spoken;
%mix.dictionaries; %mix.terminology;)' >
<!--But it defines a special version of component.seq, which
restricts each div of the text to a single base:  bases can shift only
in embedded divs or at div boundaries.  This entity is constructed out
of a series of smaller entities, one for each tag set.  If the tag set
is not in use, its entity will expand to the empty string.-->
<![%TEI.verse;[
<!--If the verse base is in use, ...-->
<!ENTITY % gen.verse '((%m.comp.verse;), (%m.common; |
%m.comp.verse;)*) |' >]]>
<![%TEI.drama;[
<!--If the drama base is in use, ...-->
<!ENTITY % gen.drama '((%m.comp.drama;), (%m.common; |
%m.comp.drama;)*) |' >]]>
<![%TEI.spoken;[
<!--If the spoken base is in use, ...-->
<!ENTITY % gen.spoken '((%m.comp.spoken;), (%m.common; |
%m.comp.spoken;)*) |' >]]>
<![%TEI.dictionaries;[
<!--If the dictionary base is in use, ...-->
<!ENTITY % gen.dictionaries '((%m.comp.dictionaries;), (%m.common; |
%m.comp.dictionaries;)*) |' >]]>
<![%TEI.terminology;[
<!--If the terminology base is in use, ...-->
```

```
<!ENTITY % gen.terminology '((%m.comp.terminology;), (%m.common; |
%m.comp.terminology;)*) |' >]]>
<!--Default declarations for all the entities gen.verse,
etc.-->
<!ENTITY % gen.verse '' >
<!ENTITY % gen.drama '' >
<!ENTITY % gen.spoken '' >
<!ENTITY % gen.dictionaries '' >
<!ENTITY % gen.terminology '' >
<!--Now we are ready to declare component.seq and
component.plus for use in general base tag set.-->
<!ENTITY % component.seq '(%m.common;)*, (%gen.verse; %gen.drama;
%gen.spoken;
%gen.dictionaries; %gen.terminology; TEI...end)?' >
<!ENTITY % component.plus '(%gen.verse; %gen.drama; %gen.spoken;
%gen.dictionaries; %gen.terminology; TEI...end)
|
( (%m.common;)+, (%gen.verse; %gen.drama; %gen.spoken;
%gen.dictionaries; %gen.terminology; TEI...end)?)' >
<!--(End of marked section for general base.)-->]]>
<!-- end of 3.7.8-->
```

3.7.9 Miscellaneous Content-Model Classes

The following element classes occupy specific places in content models; some are relevant only when certain tag sets are selected

agent elements which denote an individual or organization to whom or which responsibility for an action can be assigned

addrPart elements which can occur as part of an address

biblPart elements which can occur in bibliographic citations

demographic elements which record demographic characteristics of the participants in a text or language interaction (used in tag set for corpora and collections)

divbot elements which can occur as part of the closing material of a text division or body

divtop elements which can occur as part of the opening material of a text division or body

dramafront elements which can occur in the front matter of drama and other performance texts

front elements which can occur (at the level of text divisions) in front matter only

personPart elements which contain parts of a personal name

placePart elements which contain parts of a place name

tpParts elements which occur within title pages

fmchunk elements which can occur in place of a title page in front matter only

They are declared in the following DTD fragment:

```
<!-- 3.7.9: Misc. Element Class Models-->
<!ENTITY % x.agent "" >
<!ENTITY % m.agent "%x.agent; %n.name;">
<!ENTITY % x.addrPart "" >
<!ENTITY % m.addrPart "%x.addrPart; %n.name; | %n.postBox; | %n.postCode;
| %n.street;">
<!ENTITY % x.biblPart "" >
<!ENTITY % m.biblPart "%x.biblPart; %n.analytic; | %n.author; |
%n.biblScope; | %n.edition; | %n.editor; | %n.extent; | %n.idno; | %n.imprint; |
%n.monogr; | %n.note; | %n.pubPlace; | %n.publisher; | %n.respStmt; |
%n.series;">
<!ENTITY % x.demographic "" >
<!ENTITY % m.demographic "%x.demographic; %n.affiliation; | %n.birth; |
%n.education; | %n.firstLang; | %n.langKnown; | %n.occupation; | %n.persName; |
%n.residence; | %n.socecStatus;">
<!ENTITY % x.divbot "" >
<!ENTITY % m.divbot "%x.divbot; %n.byline; | %n.closer; | %n.dateline; |
%n.epigraph; | %n.salute; | %n.signed; | %n.trailer;">
```

```
<!ENTITY % x.divtop "" >
<!ENTITY % m.divtop "%x.divtop; %n.argument; | %n.byline; | %n.dateline; |
%n.docAuthor; | %n.docDate; | %n.epigraph; | %n.head; | %n.opener; | %n.salute; |
%n.signed;">
<!ENTITY % x.dramafront "" >
<!ENTITY % m.dramafront "%x.dramafront; %n.castList; | %n.epilogue; |
%n.performance; | %n.prologue; | %n.set;">
<!ENTITY % x.front "" >
<!ENTITY % m.front "%x.front; %n.divGen; | %m.dramafront; |
%n.titlePage;">
<!ENTITY % x.personPart "" >
<!ENTITY % m.personPart "%x.personPart; %n.addName; | %n.foreName; |
%n.genName; | %n.nameLink; | %n.roleName; | %n.surname;">
<!ENTITY % x.placePart "" >
<!ENTITY % m.placePart "%x.placePart; %n.bloc; | %n.country; |
%n.distance; | %n.geog; | %n.offset; | %n.region; | %n.settlement;">
<!ENTITY % x.tpParts "" >
<!ENTITY % m.tpParts "%x.tpParts; %n.byline; | %n.docAuthor; |
%n.docDate; | %n.docEdition; | %n.docImprint; | %n.docTitle; | %n.epigraph; |
%n.figure; | %n.imprimatur; | %n.titlePart;">
<!ENTITY % x.fmchunk "" >
<!ENTITY % m.fmchunk "%x.fmchunk; %n.argument; | %n.byline; |
%n.docAuthor; | %n.docDate; | %n.docEdition; | %n.docImprint; | %n.docTitle; |
%n.epigraph; | %n.head; | %n.titlePart;">
<!-- end of 3.7.9-->
```

3.8 Other Parameter Entities in TEI DTDs

The TEI DTDs use parameter entities for several purposes:

- to define sets of attributes shared by given classes of elements
- to define classes of elements which can occur at the same locations in content models
- to identify what base tag set should be used for a document
- to identify what additional tag sets should be included
- to include or exclude the declaration of each element
- to specify the name of each element
- to specify tag omissibility information within a DTD, or alternatively to omit such information in an XML DTD

The first two applications of parameter entities are described above in section 3.7 *Element Classes*. This chapter describes the other uses of parameter entities in the TEI DTDs.

The parameter entities used to specify which base tag set and which additional tag sets are to be used in a given document are listed in section 3.2 *Core, Base, and Additional Tag Sets*. Their default definition is always `IGNORE`: the encoder selects the TEI base and additional tag sets by declaring the appropriate parameter entities with the entity text `INCLUDE`.

The DTD and entity files are listed in section 3.2 *Core, Base, and Additional Tag Sets*; if the standard TEI entities are modified to point at locally developed DTD files containing local modifications or extensions to the TEI DTDs, the use of the standard parameter entity names ensures that the modification will be obvious upon examination of the document's DTD.

The following entities are referred to by the main tei2.dtd file to embed portions of the TEI DTDs or locally developed extensions.

TEI.extensions.ent identifies a local file containing extensions to the TEI parameter entities; see section 3.6.2 *Embedding Local Modifications*

TEI.extensions.dtd identifies a local file containing extensions to the TEI tag set; see section 3.6.2 *Embedding Local Modifications*

TEI.elementNames identifies a file containing parameter entity declarations for names of TEI elements; see section 3.8.2 *Parameter Entities for Element Generic Identifiers*

TEI.keywords identifies a file containing parameter entity declarations for TEI keywords, including the default declaration (IGNORE) of the marked-section keyword for each tag set; see section 3.8.3 *Parameter Entities for TEI Keywords*

TEI.elementClasses identifies a file containing definitions of parameter entities used in content models; see section 3.7.3 *The TEICLAS2.ENT File*

TEI.singleBase defined as INCLUDE (for normal bases) or IGNORE (for mixed and general base); used to prevent multiple definitions of the default text structure.

TEI.XML indicates whether the target DTD is to be expressed in SGML or XML. By default, this parameter entity has the value IGNORE; the user should set it to INCLUDE in order to generate an XML DTD; see 3.8.4 *Generation of an XML DTD*.

3.8.1 Inclusion and Exclusion of Elements

The TEI DTDs use marked sections and parameter entity references to allow users to exclude the definitions of individual elements, in order either to make the elements illegal in a document or to allow the element to be redefined, as further described in chapter 29 *Modifying and Customizing the TEI DTD*.

Parameter entities used for this purpose have exactly the same name as the generic identifier of the element concerned. The default definition for these parameter entities is INCLUDE but they may be changed to IGNORE in order to exclude the standard element and attribute definition list declarations from the DTD.

The declarations for the element <p>, for example, are preceded by a definition for a parameter entity with the name p and contained within a marked section whose keyword is given as %p;:

```
<!ENTITY % p 'INCLUDE' >
<![ %p; [
        <!-- element and attlist declaration for p here -->
]]>
```

These parameter entities are defined immediately preceding the element whose declarations they control; because their names are completely regular, they are not documented individually in the reference section of this document.

3.8.2 Parameter Entities for Element Generic Identifiers

In the TEI DTDs, elements are not referred to directly by their generic identifiers; instead, the DTDs refer to parameter entities which expand to the standard generic identifiers. This allows users to rename elements by redefining the appropriate parameter entity (as described more fully in chapter 29 *Modifying and Customizing the TEI DTD*). Parameter entities used for this purpose are formed by taking the standard generic identifier of the element and attaching the string "n." as a prefix. Thus the standard generic identifiers for paragraphs, notes, and quotations, <p>, <note>, and <q> are defined by declarations of the following form:

```
<!ENTITY % n.p "p">
<!ENTITY % n.pb "pb">
<!ENTITY % n.persName "persName">
```

Since all names in the TEI DTD are case-sensitive, the specific mix of upper and lower case letters in the standard generic identifier must be preserved in the entity name.

The formal declarations of the parameter entities used for generic identifiers are contained in the file teigis2.ent; since their names and replacement texts are fully predictable, these parameter entities are not individually documented in the reference section of these Guidelines. The parameter entity TEI.elementNames is used to embed the file teigis2.ent in the DTD. A full set of alternate generic identifiers can be substituted for the standard set by defining TEI.elementNames to point at a different file.[42]

[42] It is expected that after completion of the full text of these Guidelines, the TEI will prepare alternate sets of generic identifiers in languages other than English. It should be noted, however, that in the interests of simplicity parameter entities are used only for generic identifiers. Attribute names, standard attribute values, and parameter entity names are less easily modified.

3.8.3 Parameter Entities for TEI Keywords

The TEI uses the following parameter entities to signal information which cannot be expressed using attribute declaration keywords:

INHERITED indicates that an attribute value is inherited from the enclosing element, if not specified

ISO-date indicates that the attribute value should be a date in a format defined by ISO 8601:2000(E), usually yyyy-mm-dd (e.g. 1993-06-28).

extptr indicates that an attribute value should be a legal expression in the TEI extended-pointer notation

In addition, the parameter entities which control the selection of base and additional tag sets may be regarded as a keyword.

The parameter entity INHERITED is used to signal that the default value for an attribute should be inherited from an enclosing element. The definition for INHERITED is the string "#IMPLIED"; as for all implied defaults, the application program is responsible for deducing the default attribute value when no value is specified in the element start-tag. Since the parameter entity is resolved by the parser, the application program will see no difference between attributes whose default is "%INHERITED;" and those whose default is "#IMPLIED" — information about which attribute values are inherited and which are inferred in some other way must be built into the application in advance.

The parameter entity ISO-date is used to signal that the value for an attribute should be an ISO-standard date value; in this notation,[43] a date like "September 22, 1968" would be written "1968-09-22" (or alternatively as "19680922", "1968-W38-7", "1968W387", "1968-266", or "1968266"). The parameter entity ISO-date expands to "CDATA".

The keywords controlling the selection of base and additional tag sets (described in section 3.2 *Core, Base, and Additional Tag Sets*) all have the default value IGNORE; the user can override this by a local declaration, as described in section 3.3 *Invocation of the TEI DTD*.

3.8.4 Generation of an XML DTD

This version of the TEI Guidelines has been modified so that it can generate a DTD which is either SGML or XML compliant. This has been achieved by making a number of minor changes in content models of the elements[44] and by parameterising the tag omissibility indicators in each content model, since these features of SGML do not exist in XML.

Only two tag omissibility indicators are used in the TEI Guidelines: - O (i.e., the end-tag may be omitted but not the start-tag); and - - (i.e. neither start- nor end-tag may be omitted). Two parameter entities are defined, one called om.RO ("omissibility-required-optional"), and the other called om.RR ("omissibility-required-required"). By default, the value for each of these parameter entities is the appropriate indicator. Content models in the text of the Guidelines and in generated DTDs always reference the omissibility information by means of one or the other of these parameter entities.

In an XML DTD however, tags are never omissible and omissibility indicators are syntactically invalid. Thus these parameter entities are redefined as the null string, by means of a second set of declarations contained within the file teikey2.ent and controlled by a marked section guarded by the TEI.XML parameter entity. If the user declares a parameter entity TEI.XML with the value INCLUDE in the DTD subset, then the parameter entities representing tag omissibility indicators will be redefined as null strings, so that the resulting DTD is XML conformant.

---

[43] Defined by ISO 8601: 2000(E), *Data elements and interchange formats — Information interchange — Representation of dates and times* ([Geneva]: International Organization for Standardization, 2000).

[44] In general, the design goal has been to maintain backwards compatability: any document conforming to the original (P3) SGML version of the DTD should also conform to an SGML DTD described in the present document (P4), and would conform to the same DTD expressed in XML if it further followed the rules of XML (case sensitivity, always quoting attribute values, etc.). It is not, however, guaranteed that a document conforming to the present DTD will also conform to the previous one.

3.8.5 Declaration of TEI keywords

The parameter entities for TEI keywords are included in file teikey2.dtd, which is the default replacement text for the entity TEI.keywords and is embedded by the file tei2.dtd.

The file teikey2.dtd has the following contents:

```
<!-- 3.8.5: TEI Keywords-->
<!--I.  Declare TEI keywords for data types.-->
<!--These parameter entities are used as keywords to express
rules or constraints which cannot be fully expressed in attribute declarations; their
expansions show the nearest available equivalent.-->
<!ENTITY % INHERITED '#IMPLIED' >
<!ENTITY % ISO-date 'CDATA' >
<!ENTITY % extPtr 'CDATA' >
<!--II.  Declare keywords for tag-set selection.-->
<!--Declare all bases and additional tag sets as IGNORE.  The
user can override this default by declaring the same entity with the
replacement text INCLUDE, in the document's DTD
subset.-->
<!--Base tag sets first.-->
<!ENTITY % TEI.prose 'IGNORE' >
<!ENTITY % TEI.verse 'IGNORE' >
<!ENTITY % TEI.drama 'IGNORE' >
<!ENTITY % TEI.spoken 'IGNORE' >
<!ENTITY % TEI.dictionaries 'IGNORE' >
<!ENTITY % TEI.terminology 'IGNORE' >
<!--Now the mixed bases.-->
<!ENTITY % TEI.general 'IGNORE' >
<!ENTITY % TEI.mixed 'IGNORE' >
<!--Now additional tag sets.-->
<!ENTITY % TEI.linking 'IGNORE' >
<!ENTITY % TEI.analysis 'IGNORE' >
<!ENTITY % TEI.fs 'IGNORE' >
<!ENTITY % TEI.certainty 'IGNORE' >
<!ENTITY % TEI.transcr 'IGNORE' >
<!ENTITY % TEI.textcrit 'IGNORE' >
<!ENTITY % TEI.names.dates 'IGNORE' >
<!ENTITY % TEI.nets 'IGNORE' >
<!ENTITY % TEI.figures 'IGNORE' >
<!ENTITY % TEI.corpus 'IGNORE' >
<!--III. Declare TEI.XML and associated omissibility indicators-->
<!ENTITY % TEI.XML 'IGNORE' >
<![%TEI.XML;[
<!ENTITY % om.RO '' >

<!ENTITY % om.RR '' >]]>
<!ENTITY % om.RO '- O' >
<!ENTITY % om.RR '- -' >
<!-- end of 3.8.5-->
```

The relevant portion of the main DTD looks like this:

```
<!-- 3.8.5: TEI Keywords-->
<!--We declare and immediately embed the TEI keywords file.-->
<!ENTITY % TEI.keywords.ent PUBLIC '-//TEI P4//ENTITIES TEI
Keywords//EN' 'teikey2.ent' >%TEI.keywords.ent;
<!-- end of 3.8.5-->
```

II: Core Tags and General Rules