**The Association for Computers and the Humanities (ACH)**

**The Association for Computational Linguistics (ACL)**

**The Association for Literary and Linguistic Computing (ALLC)**

# Guidelines for Electronic Text Encoding and Interchange

Edited by

C. M. Sperberg-McQueen and Lou Burnard

TEI P2, Chapter 3
Text Encoding Initiative
Chicago, Oxford



April 7, 1993

Draft Version 2, April 7, 1993

# *Note*

This is the current draft (as of April 7, 1993) of one section of the *Guidelines for Electronic Text Encoding and Interchange*, version 2 (also known as TEI P2), produced by the Text Encoding Initiative. This fascicle contains:

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

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

TEI P2, like P1, is a working draft. The observant will find several typographic and presentational flaws in the current text: for example, cross-references are not properly resolved in the LaTeX versions and some parts of the reference material are not yet complete. We hope to rectify as many as possible of these flaws in the coming months, but have decided not to allow them to delay any further the release of the draft. We hope that those interested in the Guidelines will find the current state of the text readable and usable enough for the purpose of substantive comment. As far as possible, all comments received on this section will be taken into account in the preparation of the next version of the Guidelines, which is due by the end of 1992.

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

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

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

- TEI ED J8, “Obtaining the Second Version of the TEI Guidelines (TEI P2)” (describes how to retrieve electronic copies of TEI P2 and the various formats they are available in)

- TEI ED J9, “Obtaining Paper Copies of the Second Version of the TEI Guidelines (TEI P2)” (describes how to request paper copies of TEI P2, for those without electronic mail access)

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

This document is made possible in part by financial support from the U.S. National Endowment for the Humanities, an independent federal agency; Directorate General XIII of the Commission of the European Communities; the Andrew W. Mellon Foundation; and the Social Science and Humanities Research Council of Canada. Direct and indirect support has also been received from the University of Illinois at Chicago, the Oxford University Computing Service, the University of Arizona, the University of Oslo, Queen's University (Kingston, Ont.), and Ohio State University. We acknowledge assistance with software acquisition from the following firms: SEMA Group and Yard Software (for Mark-It and Write-It), Software Exoterica (for CheckMark and Xtran), SoftQuad, Inc., (for Author/Editor and RulesBuilder), Electronic Book Technologies (for demonstration copies of DynaText), Xerox Corporation (for Ventura Publisher).

# *Overall Table of Contents for TEI P2*

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

# *Contents*

# Chapter 3

## *Structure of the TEI Document Type Declarations*

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 in SGML. It assumes some familiarity with SGML; see chapter 2, “Concise Summary of SGML,” [in separate fascicle].

The TEI encoding scheme consists of 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, in order to form a TEI *main DTD*. Auxiliary tag sets are also defined for specific purposes independent of the main TEI 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 discusses these distinctions, and list the specific tag sets included in each category. Section 3.3, “Invocation of TEI DTDS,” on page 7 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,” on page 10.

The remainder of the chapter contains a more technical description of the SGML 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 section 32, “Modifying TEI DTDs,” [in separate fascicle]), 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,” on page 14. The *element classes* used to define smaller groups of elements and their characteristics are described in section 3.7, “Element Classes,” on page 22. Both global attributes and element classes are implemented using SGML *parameter entities*; various other uses of parameter entities in the TEI DTDs are discussed in section 3.8, “Other Parameter Entities in TEI DTDs,” on page 36.

## 3.1 Main and Auxiliary DTDs

These Guidelines define a large number of SGML tags 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. The tags are grouped into *tag sets* or *DTD fragments*, each comprising a set of declarations for tags which belong together in some respect, typically related to their intended application area.

All tags 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," on page 3 below.

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:

**structured 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," [in separate fascicle] and 27, "Structured Header," [in separate fascicle]).

**writing system declaration**: used to define and document character sets or transliteration schemes (see chapters 4, "Characters and Character Sets," [in separate fascicle]and 28, "Writing System Declaration," [in separate fascicle]).

**feature system declaration**: used to define and document sets of analytic features (see chapters 18, "Feature Structure Analysis," [in separate fascicle] and 29, "Feature System Declaration," [in separate fascicle]).

**tag set declaration**: used to define and document descriptive documentation for TEI-conformant tag sets (see chapter 30, "Tag Set Declaration," [in separate fascicle]).

A structured 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 34, "Use of TEI Encoding Scheme in Interchange," [in separate fascicle].

The DTD files containing these auxiliary DTDs are:

*teishd2.dtd* : structured 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 all TEI main DTDs; 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 invoked as a unit by appropriate SGML *parameter entity declarations*.[1] Several such declarations may be needed to invoke all parts of a given tag set, since as well as defining elements or attributes, a tagset 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*.
**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**: if *xxx* is a base tag set, this contains a list of any component-level elements unique to it (for a definition of component-level elements, see section 3.7, "Element Classes," on page 22)

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

[1] A *parameter entity* is an SGML construct; for further explanation, review chapter 2, "Concise Summary of SGML," [in separate fascicle].

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 in section 3.6, "The TEI2.DTD File," on page 14 below. 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 version 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, "Tags Available in All TEI DTDs," [in separate fascicle]

*TEI.header.dtd* : refers to the file *teihdr2.dtd*, which declares the tags of the TEI header defined in chapter 5, "The TEI Header," [in separate fascicle]

Together with these tagsets, part II also documents a tag set for default text structure, and for 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 chapter 14, "Composite Texts and Combining Bases," [in separate fascicle]. 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 SGML 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 SGML 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*
*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.letters* : selects the base tag set for letters and memos, contained in *teilett2.dtd* and *teilett2.ent*
*TEI.dictionaries* : selects the base tag set for print dictionaries; *teidict2.dtd* and *teidict2.ent*
*TEI.terminology* : selects the base tag set for terminological data files; *teiterm2.dtd* and *teiterm2.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 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,” on page 14.

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>**, **<div>**, 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,” [in separate fascicle].

### 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 16, “Segmentation and Alignment,” [in separate fascicle])

*TEI.analysis* : embeds the files *teiana2.dtd* and *teiana2.ent*, with tags for simple analytic mechanisms (chapter 17, “Simple Analytic Mechanisms,” [in separate fascicle])

*TEI.fs* : embeds the files *teifs2.dtd* and *teifs2.ent*, with tags for feature structure analysis (chapter 18, “Feature Structure Analysis,” [in separate fascicle])

*TEI.certainty* : embeds the files *teicert2.dtd* and *teicert2.ent*, with tags for indicating uncertainty and probability in the markup (chapter 19, “Certainty,” [in separate fascicle])

*TEI.physical* : embeds the file *teiphys2.dtd*, with tags for manuscripts, analytic bibliography, and physical description (chapter 20, “Manuscripts, Analytic Bibliography, and Physical Description of the Source Text,” [in separate fascicle])

*TEI.textcrit* : embeds the file *teitc2.dtd*, with tags for critical editions (chapter 21, “Text Criticism and Apparatus,” [in separate fascicle])

*TEI.names.dates* : embeds the file *teind2.dtd*, with tags for specialized tags for names and dates (chapter 22, “Additional Tags for Names and Dates,” [in separate fascicle])

*TEI.nets* : embeds the file *teinet2.dtd*, with tags for graphs, digraphs, trees, and other networks (chapter 23, “Graphs, Digraphs, and Trees,” [in separate fascicle]) — not to be confused with the graphics markup of *TEI.figures*

*TEI.figures* : embeds the file *teifig2.dtd*, with tags for graphics, figures, and illustrations (chapter 24, “Graphics, Figures, and Illustrations,” [in separate fascicle]) — not to be confused with the graph-theoretic markup of *TEI.nets*

*TEI.formulas* : embeds the file *teiform2.dtd*, with tags for formulæ and tables (chapter 25, “Formulae and Tables,” [in separate fascicle])

*TEI.corpus* : embeds the file *teicorp2.dtd*, with tags for additional tags for language corpora (chapter 26, “Additional Tag Set for Language Corpora,” [in separate fascicle])

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 feature structures, for example, is enabled by declaring the parameter entity *TEI.fs* as “INCLUDE”; this in turn enables declarations of *TEI.fs.ent* and *TEI.fs.dtd* as the system files *teifs2.ent* and *teifs2.dtd*.

### 3.2.4 User-Defined Tag Sets

As described in chapter 32, "Modifying TEI DTDs," [in separate fascicle], 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, 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," on page 17.

Further recommendations for the creation of user-defined extension or modification are provided in chapters 32, "Modifying TEI DTDs," [in separate fascicle] and 31, "TEI Conformance," [in separate fascicle].

## 3.3 Invocation of TEI DTDS

Like any other SGML document, a TEI document must begin with a document type declaration (DTD). Local systems may allow the DTD to be implicit, but for interchange purposes it *must* be explicit. Because of its highly modular nature, it may in any case be desirable for the component parts of a 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," on page 3. For example, a document using the base tag set for prose will begin with a document type declaration something like this:

```
<!DOCTYPE TEI.2 system 'tei2.dtd' [
     <!ENTITY % TEI.prose 'INCLUDE' >
]>
```

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

```
<!DOCTYPE TEI.2 system 'tei2.dtd' [
     <!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 graphics, for example, will begin with a document type declaration something like this:

```
<!DOCTYPE TEI.2 system 'tei2.dtd' [
<!-- TEI base tag set specified here: ... -->
     <!ENTITY % TEI.prose 'INCLUDE' >

<!-- TEI additional tag sets optionally specified here: ... -->
     <!ENTITY % TEI.textcrit 'INCLUDE' >
     <!ENTITY % TEI.graphics '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 system 'tei2.dtd' [
<!-- 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.graphics '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 26, “Additional Tag Set for Language Corpora,” [in separate fascicle])
- 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,” [in separate fascicle])
- a notion of *embedded text* which allows one text to be embedded within another

Whichever mechanism is adopted, if the whole of the resulting document is to be parseable by a single 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 system 'tei.2' [
  <!ENTITY % TEI.mixed 'INCLUDE' >
  <!ENTITY % TEI.prose 'INCLUDE' >
  <!ENTITY % TEI.drama 'INCLUDE' >
  <!ENTITY % TEI.dictionary 'INCLUDE' >
  <!-- Moby Dick has some unusual structures, for a 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 **<div>** 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 system 'tei.2' [
  <!ENTITY % TEI.general 'INCLUDE' >
  <!ENTITY % TEI.prose 'INCLUDE' >
  <!ENTITY % TEI.drama 'INCLUDE' >
  <!ENTITY % TEI.dictionary 'INCLUDE' >
]>
```

For further discussion of mixing TEI bases, see chapter 14, "Composite Texts and Combining Bases," [in separate fascicle].

## 3.5 Global Attributes

The following attributes are defined for every TEI element.[2]

*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," on page 22.

An additional attribute *TEIform* attribute is also defined for every TEI element. Because its default value is different in every case, this attribute cannot be inherited from the global element class but must be defined individually for each element.[3]

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

---

[2] 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 they may be inherited and over-ridden, see section 3.7.1, "Classes Which Share Attributes," on page 22.

[3] However, a dummy element class *TEIform* is defined in the reference section, solely for documentary purposes.

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>
```

ID values must be legal SGML names; by default, this means they must begin with a letter from A to Z or a to z and contain no characters other than the letters A to Z or a to z, the digits 0 to 9, the full stop, and the hyphen. Furthermore, by default upper and lower case letters are not distinguished: thus, the strings “a23” and “A23” are identical, and may not be used to identify two distinct elements. If two elements are given the same identifier, the SGML 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,” [in separate fascicle].

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:

```
<list type=ordered>
<item n=1>list item 1
<item n=2>list item 2
<item n=2>list item 3, misnumbered '2' in the original
<item n=4>list item 4
</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 examples:

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

<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, “Characters and Character Sets,” [in separate fascicle].

```
<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.[4] 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,” [in separate fascicle].

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:

&@fn@sep

[4] SGML 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, that the *lang* attribute point to a **<language>** element.

```
<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 for all and specify a *rend* value only for any elements which deviate from the usual rendition.[5]

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 TEI header.[6]

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.[7]

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 32, “Modifying TEI DTDs,” [in separate fascicle], 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.

If elements are not just renamed but redefined, the *TEIform* attribute can be given an appropriate value to indicate the corresponding TEI element; similarly, new elements may be identified as subclasses of TEI elements by giving an appropriate default value to *TEIform*.[8]

Constraints on the use of *TEIform* (rules determining when it is legitimate, and when not, to identify a new element as corresponding to a given TEI element) are given in chapter 32, “Modifying TEI DTDs,” [in separate fascicle]. Further discussion of the relationship between the TEI DTD and other standard DTDs is given in chapter 35, “Relationship of TEI to Other Standards,” [in separate fascicle].

&@fn@sep

[5] The current draft of the TEI header provides no location for this information; this will be amended in the next revision.

[6] Again, no location for this information is yet provided for in the TEI Header.

[7] The *TEIform* attribute is based on the notion of *architectural forms* developed for HyTime (ISO 10744).

[8] As currently published, TEI DTD fragments are yet to be be modified to include declarations for TEIform on all elements.

The formal definition of the global attributes is as follows:

```
<!-- 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 are inherited by GLOBAL; otherwise the  -->
<!-- parameter entities referred to expand to the empty       -->
<!-- string.                                                   -->


<!ENTITY % a.global '        %a.spoken;
                             %a.certainty;
                             %a.fs;
                             %a.analysis;
                             %a.linking;
                             %a.terminology;
          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.                                       -->

<!-- This fragment is used in sec. 3.7.3                       -->
```

## 3.6 The TEI2.DTD File

All TEI-encoded documents use the same top-level DTD file, into which other DTD files are embedded, 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:
   a. declares TEI-specific keywords used in other declarations (see section 3.8.3, “Parameter Entities for TEI Keywords,” on page 38)
   b. declares parameter entities for TEI generic identifiers (by embedding the file *teigis2.ent*; see section 3.8.2, “Parameter Entities for Element Generic Identifiers,” on page 38)
   c. declares default values of “IGNORE” for all the parameter entities used to select base and additional tag sets

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,” on page 26)
3. declares the top-level elements **<TEI.2>** and **<TEI.corpus.2>**
4. embeds DTD files containing local modifications, 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 version of the entity *component* which applies to 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: Guidelines for Electronic      -->
<!-- Text Encoding and Interchange. DRAFT Version 2. 1992-93. -->

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

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


<!-- 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    -->
<!-- TEI.2.corpus elements.                                   -->


<!-- I.  Preliminaries.                                       -->

<!-- ... declarations from section 3.6.2                      -->
<!--     (Local modifications to parameter entities)          -->
<!--     go here ...                                          -->

<!-- Define entities for TEI keywords.                        -->
```

```
<!-- ... declarations from section 3.8.3                    -->
<!--     (TEI Keywords)                                     -->
<!--     go here ...                                        -->

<!-- Embed entities for TEI generic identifiers.            -->

<!ENTITY % TEI.elementNames system 'teigis.ent' >
%TEI.elementNames;

<!-- 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.letters '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.physical 'IGNORE' >
<!ENTITY % TEI.textcrit 'IGNORE' >
<!ENTITY % TEI.names.dates 'IGNORE' >
<!ENTITY % TEI.nets 'IGNORE' >
<!ENTITY % TEI.figures 'IGNORE' >
<!ENTITY % TEI.formulas 'IGNORE' >
<!ENTITY % TEI.corpus 'IGNORE' >

<!-- 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 system '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          - O  (TeiHeader, text)                >
<!ATTLIST TEI.2               %a.global;                       >

<!-- A TEI corpus is a series of TEI.2 documents, preceded   -->
<!-- by a corpus-level TEI header.                           -->

<!ELEMENT TEI.corpus.2   - O  (TeiHeader, tei.2+)              >
<!ATTLIST TEI.corpus.2        %a.global;                       >

<!-- 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 section 3.6.2                     -->
<!--     (Embed local element declarations, etc.)            -->
<!--     go here ...                                         -->
<!-- ... declarations from section 3.6.3                     -->
<!--     (Embed the core tag sets)                           -->
<!--     go here ...                                         -->
<!-- ... declarations from section 3.6.4                     -->
<!--     (Embed base tag set)                                -->
<!--     go here ...                                         -->
<!-- ... declarations from section 3.6.5                     -->
<!--     (Embed additional tag sets)                         -->
<!--     go here ...                                         -->
```

A TEI-conformant document *must* use the *tei2.dtd* file; 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 TEI DTDS,” on page 7.

### 3.6.2 Embedding Local Modifications

As noted above in section 3.2.4, “User-Defined Tag Sets,” on page 7, 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 SGML 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 32, “Modifying TEI DTDs,” [in separate fascicle], 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;
<!-- This fragment is used in sec. 3.6.1                      -->
```

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 the use of the standard entity references for phrase sequence and global attributes):

```
<!ELEMENT it          - -  (%phrase.seq)                       >
<!ATTLIST it               %a.global
          rend             CDATA          #FIXED 'italics'    >
<!ELEMENT bd          - -  (%phrase.seq)                       >
<!ATTLIST bd               %a.global
          rend             CDATA          #FIXED 'boldface'   >
```

For further examples of local modifications to both parameter entities and element declarations, see chapter 32, “Modifying TEI DTDs,” [in separate fascicle].

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;
<!-- This fragment is used in sec. 3.6.1                      -->
```

### 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 system 'teihdr2.dtd' >
%TEI.header.dtd;
<!ENTITY % TEI.core.dtd system 'TEIcore2.dtd' >
%TEI.core.dtd;
<!-- This fragment is used in sec. 3.6.1                      -->
```

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,” on page 3. 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 system 'teipros2.dtd' >
%TEI.prose.dtd;
]]>

<![ %TEI.verse [
<!ENTITY % TEI.verse.dtd system 'teivers2.dtd' >
%TEI.verse.dtd;
]]>

<![ %TEI.drama [
<!ENTITY % TEI.drama.dtd system 'teidram2.dtd' >
%TEI.drama.dtd;
]]>

<![ %TEI.spoken [
<!ENTITY % TEI.spoken.dtd system 'teispok2.dtd' >
%TEI.spoken.dtd;
]]>

<![ %TEI.letters [
<!ENTITY % TEI.letters.dtd system 'teilett2.dtd' >
%TEI.letters.dtd;
]]>

<![ %TEI.dictionary [
<!ENTITY % TEI.dictionaries.dtd system 'teidict2.dtd' >
%TEI.dictionary.dtd;
]]>

<![ %TEI.terminology [
<!ENTITY % TEI.terminology.dtd system 'teiterm2.dtd' >
%TEI.terminology.dtd;
]]>

<![ %TEI.general [
<!ENTITY % TEI.general.dtd system 'teigen2.dtd' >
%TEI.general.dtd;
]]>

<![ %TEI.mixed [
<!ENTITY % TEI.mixed.dtd system 'teimix2.dtd' >
%TEI.mixed.dtd;
]]>

<!-- This fragment is used in sec. 3.6.1                      -->
```

### 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," on page 3. 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 system 'teilink2.dtd' >
%TEI.linking.dtd;
]]>

<![ %TEI.analysis [
<!ENTITY % TEI.analysis.dtd system 'teiana2.dtd' >
%TEI.analysis.dtd;
]]>

<![ %TEI.fs [
<!ENTITY % TEI.fs.dtd system 'teifs2.dtd' >
%TEI.fs.dtd;
]]>

<![ %TEI.certainty [
<!ENTITY % TEI.certainty.dtd system 'teicert2.dtd' >
%TEI.certainty.dtd;
]]>

<![ %TEI.physical [
<!ENTITY % TEI.physical.dtd system 'teiphys2.dtd' >
%TEI.physical.dtd;
]]>

<![ %TEI.textcrit [
<!ENTITY % TEI.textcrit.dtd system 'teitc2.dtd' >
%TEI.textcrit.dtd;
]]>

<![ %TEI.names.dates [
<!ENTITY % TEI.names.dates.dtd system 'teind2.dtd' >
%TEI.names.dates.dtd;
]]>

<![ %TEI.nets [
<!ENTITY % TEI.nets.dtd system 'teinet2.dtd' >
%TEI.nets.dtd;
]]>
```

```
<![ %TEI.figures [
<!ENTITY % TEI.figures.dtd system 'teifig2.dtd' >
%TEI.figures.dtd;
]]>

<![ %TEI.formulas [
<!ENTITY % TEI.formulas.dtd system 'teiform2.dtd' >
%TEI.formulas.dtd;
]]>

<![ %TEI.corpus [
<!ENTITY % TEI.corpus.dtd system 'teicorp2.dtd' >
%TEI.corpus.dtd;
]]>

<!-- This fragment is used in sec. 3.6.1                      -->
```

## 3.7 Element Classes

The TEI DTD contains over two 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 SGML 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, or 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,” on page 36.

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 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>**, **<place>**, or **<person>**. 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.[9]

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'      >
```

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          - -  (%phrase.seq;)                    >
<!ATTLIST name               %a.global;
                             %a.names;
          type              CDATA               #IMPLIED       >
```

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;            #IMPLIED
          to                %extPtr;            "DITTO"'       >
```

(For an explanation of the parameter entity *extptr* used in the above example, see section 3.8.3, "Parameter Entities for TEI Keywords," on page 38.)

The a-classes declared in these Guidelines are:[10]

---

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

[10] This list includes all a-classes defined at the time this chapter was being completed; others may be added as the current draft is being completed. -Ed.

**declaring**: elements which have a *decls* attribute for specifying which declarations in the header apply to the element, as described in section 26.3, “Associating Contextual Information with a Text,” [in separate fascicle]
**declarable**: header elements containing declarations, which can be pointed at by the *decls* attribute, as described in section 26.3, “Associating Contextual Information with a Text,” [in separate fascicle]
**divn**: structural elements which behave in the same way as divisions, as described in section 7.1, “Divisions of the Body,” [in separate fascicle].
**flatterm**: elements used in the “flat” style of terminological data, which take specialized attributes for indicating logical dependencies, as described in section 13.3.3, “Flat Term Entries Using Group and Depend Attributes,” [in separate fascicle]
**names**: elements which contain proper nouns and share attributes for identifying their referents and regularizing their spelling (section 6.4.1, “Names,” [in separate fascicle])
**pointer**: elements which point from one location in the document to another (section 6.6, “Simple Cross References,” [in separate fascicle])
**timed**: elements (in the base tag set for spoken texts) which have a duration in time expressible with the attributes, as described in section 10.4.3.2, “Temporal Information,” [in separate fascicle]
**xPointer**: elements which point from one location in the document to other locations within or outside the current document (section 16.2, “External pointers and references,” [in separate fascicle])

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.spoken**: additional global attributes for the spoken-text base
**a.terminology**: additional global attributes for the terminology base
**a.linking**: additional global attributes for the linking tag set
**a.analysis**: additional global attributes for the analysis tag set
**a.fs**: additional global attributes for the feature structure tag set
**a.certainty**: additional global attributes for the uncertainty tag set

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,” on page 30. For purposes of documentation, these attributes are treated as if inherited by the class *global* from superclasses called *spoken*, *terminology*, etc., and are documented under the class name.

One further complication to the inheritance mechanism should be mentioned here. In rare cases, a member of an a-class may override the definition of an inherited attribute. For example, the element **<anchor>** inherits the global *id* attribute from the class *global* -- as does every other element. On **<anchor>** elements, however, *id* is not optional but required. The declaration of **<anchor>** therefore does not refer to the class *global*, but instead defines all the inherited attributes explicitly, using its own declaration for *id* and the default inherited declarations for the other global attributes:

```
<!-- 3.7.1: Sample of local declaration overriding inherited  -->
<!-- attributes                                               -->
<!-- ANCHOR overrides the default ID, but uses the default    -->
<!-- declarations for the other global attributes.            -->

<!ELEMENT anchor        - O  EMPTY                              >
<!ATTLIST anchor
          n                  CDATA              #IMPLIED
          lang               IDREF              %INHERITED
          rend               CDATA              #IMPLIED
          id                 ID                 #REQUIRED       >
```

Because this declaration does not use the parameter entity *a.global*, clearly any change in the definition for that entity will not be reflected in this declaration. Consequently, any changes made to the global attributes as such will not be inherited by the **<anchor>** element. Instead, such changes must be replicated manually. Care must thus be taken in modifying attribute definitions for a-classes if any members of the class override the inherited definitions, to ensure that all members of the class really do get the modified definitions.

### 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, this fact is indicated by grouping them 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 SGML 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 the 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>**, **<bibl.full>**, and **<bibl.struct>**. Its content-model entity is defined thus:

```
<!ENTITY % x.bibl '' >
<!ENTITY % m.bibl '%x.bibl bibl | bibl.full | bibl.struct' >
```

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

```
<!ENTITY % x.bibl 'myBib |' >
```

This changes the replacement text of *m.bibl* from its default value to “myBib | bibl | bibl.full | bibl.struct”. 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.hqphrase; | %m.loc; | %m.seg;' >
```

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 three-fold 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. **<div>** 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 **<div>**)[11]

&@fn@sep

[11] 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

*inter-level elements* : elements such as lists, notes, quotations, etc. which can appear either between chunks (as children of a **<div>**) 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"

In general, the body of any text comprises a series of components, optionally grouped into **<div>** 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: Guidelines for Electronic     -->
<!-- Text Encoding and Interchange. DRAFT Version 2. 1992-93. -->

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

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


<!-- 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 section 3.7.4                     -->
<!--     (Low-level classes)                                 -->
<!--     go 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    -->
```

restrictive sense.

```
<!-- inter-level elements), as opposed to the                -->
<!-- tagset-specific components.                              -->


<!-- ... declarations from section 3.7.5                      -->
<!--     (Common high-level classes)                          -->
<!--     go 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 section 3.7.6                      -->
<!--     (Embedding tag-set-specific entity definitions)      -->
<!--     go here ...                                          -->

<!-- We can now declare the standard content models; one of   -->
<!-- these varies with the base selected.                     -->

<!-- ... declarations from section 3.7.7                      -->
<!--     (Standard Content Models)                            -->
<!--     go here ...                                          -->

<!-- Finally, we declare the global attributes.               -->

<!-- ... declarations from section 3.5                        -->
<!--     (Global attributes)                                  -->
<!--     go here ...                                          -->
```

### 3.7.4 Low-Level Element Classes

The following low-level classes group together sets of semantically or structurally similar elements.[12] 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 Marks,” [in separate fascicle]

&@fn@sep

[12] This list includes all m-classes defined at the time this chapter was being completed; others may be added as the current draft is being completed. -Ed.

*data* : elements for recording information about the referents of a text, including those defined in section 6.4, “Names, Numbers, Dates, and Abbreviations,” [in separate fascicle]
*edit* : elements for recording simple editorial interventions in a text, including those defined in section 6.5, “Simple Editorial Changes,” [in separate fascicle]
*loc* : elements for recording location information in a text, including those defined in section 6.9, “Reference Systems,” [in separate fascicle]

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 Marks,” [in separate fascicle]
*bibl* : elements for bibliographic citations; see section 6.10, “Bibliographic Citations,” [in separate fascicle]
*lists* : elements for lists; see section 6.7, “Lists,” [in separate fascicle]
*notes* : general-purpose annotation elements; see section 6.8, “Notes,” [in separate fascicle]

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 distinct | emph | foreign |
           gloss | hi | mentioned | soCalled | term' >
<!ENTITY % x.data '' >
<!ENTITY % m.data '%x.data abbr | address | date | dateRange |
           expan | measure | name | num | org | organization |
           person | place | time | timeRange' >
<!ENTITY % x.edit '' >
<!ENTITY % m.edit '%x.edit add | corr | del | omit | orig | reg
           | sic' >
<!ENTITY % x.loc '' >
<!ENTITY % m.loc '%x.loc corresp | correspGrp | lb | link | pb
           | ptr | ref | xptr | xref' >
<!ENTITY % x.seg '' >
<!ENTITY % m.seg '%x.seg s' >

<!-- Inter-level classes                                    -->

<!ENTITY % x.hqinter '' >
<!ENTITY % m.hqinter '%x.hqinter q | quote' >
<!ENTITY % x.bibl '' >
<!ENTITY % m.bibl '%x.bibl bibl | bibl.full | bibl.struct' >
<!ENTITY % x.lists '' >
<!ENTITY % m.lists '%x.lists label | list | list.bibl' >
```

```
<!ENTITY % x.notes '' >
<!ENTITY % m.notes '%x.notes note | span' >
<!-- This fragment is used in sec. 3.7.3                          -->
```

### 3.7.5 High-Level Element Classes

The following element classes are used to implement the three-fold structural distinction among phrases, chunks, and intermediate elements discussed above in section 3.7.3, “The TEICLAS2.ENT File,” on page 26. In this terminology, *chunk* 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 chapter 14, “Composite Texts and Combining Bases,” [in separate fascicle].

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.hqphrase; | %m.loc; | %m.seg;' >
<!ENTITY % x.inter '' >
<!ENTITY % m.inter '%x.inter %m.bibl; | %m.hqinter; | %m.lists;
           | %m.notes; | text' >
<!ENTITY % x.chunk '' >
<!ENTITY % m.chunk '%x.chunk l | lg | p | sp | stage' >

<!-- 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;' >
<!-- This fragment is used in sec. 3.7.3                          -->
```

### 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.letters* : elements unique to letters and memoranda
*comp.dictionary* : 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 system 'teivers2.ent' >
%TEI.verse.ent;
]]>

<![ %TEI.drama [
<!ENTITY % TEI.drama.ent system 'teidram2.ent' >
%TEI.drama.ent;
]]>

<![ %TEI.spoken [
<!ENTITY % TEI.spoken.ent system 'teispok2.ent' >
%TEI.spoken.ent;
]]>

<![ %TEI.letters [
<!ENTITY % TEI.letters.ent system 'teilett2.ent' >
%TEI.letters.ent;
]]>

<![ %TEI.dictionaries [
<!ENTITY % TEI.dictionaries.ent system 'teidict2.ent' >
%TEI.dictionaries.ent;
]]>

<![ %TEI.terminology [
<!ENTITY % TEI.terminology.ent system 'teiterm2.ent' >
%TEI.terminologg.ent;
]]>

<![ %TEI.linking [
<!ENTITY % TEI.linking.ent system 'teilink2.ent' >
%TEI.linking.ent;
]]>

<![ %TEI.analysis [
<!ENTITY % TEI.analysis.ent system 'teiana2.ent' >
%TEI.analysis.ent;
]]>
```

```
<![ %TEI.fs [
<!ENTITY % TEI.fs.ent system 'teifs2.ent' >
%TEI.fs.ent;
]]>

<![ %TEI.certainty [
<!ENTITY % TEI.certainty.ent system 'teicert2.ent' >
%TEI.certainty.ent;
]]>

<!-- This fragment is used in sec. 3.7.3                        -->
```

### 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)' >
<!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 section 3.7.8                      -->
<!--     (Definition of components for combined bases)        -->
<!--     go here ...                                          -->

<![ %TEI.prose [
<!ENTITY % component '(%m.common)' >
]]>

<![ %TEI.verse [
<!ENTITY % component '(%m.common | %m.comp.verse)' >
]]>

<![ %TEI.drama [
<!ENTITY % component '(%m.common | %m.comp.drama)' >
]]>

<![ %TEI.spoken [
<!ENTITY % component '(%m.common | %m.comp.spoken)' >
]]>

<![ %TEI.letters [
<!ENTITY % component '(%m.common | %m.comp.letters)' >
]]>

<![ %TEI.prose [
<!ENTITY % component '(%m.common)' >
]]>

<![ %TEI.dictionaries [
<!ENTITY % component '(%m.common | %m.comp.dictionaries)' >
]]>

<![ %TEI.terminology [
<!ENTITY % component '(%m.common | %m.comp.terminology)' >
]]>

<!-- 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;)*' >

<!-- The following entities do not vary with the base.        -->

<!ENTITY % paraContent '(#PCDATA | %m.phrase | %m.inter)*' >
<!ENTITY % specialPara '(((%m.chunk), (%component.seq))
           | (%paraContent))' >
```

```
<!-- This fragment is used in sec. 3.7.3                      -->
```

### 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      -->
<![ %TEI.mixed [
<!-- 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.letters %mix.dictionaries
           %mix.terminology' >
]]>

<![ %TEI.general [
<!-- The general base uses the same definition of component   -->
<!-- as the mixed base.                                       -->

<!ENTITY % component '(%m.common %mix.verse %mix.drama
           %mix.spoken %mix.letters %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.letters [
<!-- If the letters base is in use, ...                         -->

<!ENTITY % gen.letters '((%m.comp.letters),
           (%m.common | %m.comp.letters)*) |' >
]]>

<![ %TEI.dictionaries [
<!-- If the dictionary base is in use, ...                      -->

<!ENTITY % gen.dictionary '((%m.comp.dictionary),
           (%m.common | %m.comp.dictionary)*) |' >
]]>

<![ %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.letters '' >
<!ENTITY % gen.dictionary '' >
<!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.letters
  %gen.dictionary
  %gen.terminology
  TEI...end
)?'
 >
<!ENTITY % component.plus '(%gen.verse
  %gen.drama
  %gen.spoken
  %gen.letters
  %gen.dictionary
```

```
  %gen.terminology
  TEI...end
) |
( (%m.common)+, (
  %gen.verse
  %gen.drama
  %gen.spoken
  %gen.letters
  %gen.dictionary
  %gen.terminology
  TEI...end
)?'
 >

<!-- (End of marked section for general base.)                -->

]]>


<!-- This fragment is used in sec. 3.7.7                      -->
```

### 3.7.9 Miscellaneous Content-Model Classes

The following element classes occupy specific places in content models:

**biblpart**: elements which can occur in bibliographic citations
**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
**refsys**: milestone elements used in reference systems

They are declared in the following DTD fragment:

```
<!-- 3.7.9:  Misc. Element Class Models                       -->
<!ENTITY % x.bibl.part '' >
<!ENTITY % m.bibl.part '%x.bibl.part analytic | author |
           biblScope | city | edition | editor | extent | idno
           | imprint | monogr | note | publisher | resp |
           series' >
<!ENTITY % x.divtop '' >
<!ENTITY % m.divtop '%x.divtop byline | docAuthor | docDate |
           epigraph | head | salute | signed' >
<!ENTITY % x.divbot '' >
<!ENTITY % m.divbot '%x.divbot byline | epigraph | salute |
           signed | trailer' >
<!ENTITY % x.refsys '' >
<!ENTITY % m.refsys '%x.refsys milestone' >
```

## 3.8 Other Parameter Entities in TEI DTDs

The TEI DTDs use SGML 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

The first two applications of parameter entities are described above in section 3.7, “Element Classes,” on page 22. 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,” on page 3. 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,” on page 3; 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,” on page 17

*TEI.extensions.dtd* : identifies a local file containing extensions to the TEI tag set; see section 3.6.2, “Embedding Local Modifications,” on page 17

*TEI.elementClasses* : identifies a file containing definitions of parameter entities used in content models; see section 3.7.3, “The TEICLAS2.ENT File,” on page 26

*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,” on page 38

### 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 32, “Modifying TEI DTDs,” [in separate fascicle].

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 32, "Modifying TEI DTDs," [in separate fascicle]). Parameter entities used for this purpose are formed by taking the standard name (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.note 'note' >
<!ENTITY % n.q    'q' >
```

Since by default parameter entities are case-sensitive, the specific mix of upper and lower case letters in the standard name must be preserved in the entity name.

The formal declarations of the parameter entities used for generic identifiers are contained in the file *teigis.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 *teigis.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.[13]

---

[13] 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,

### 3.8.3 Parameter Entities for TEI Keywords

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

**INHERITED**: indicates that an attribute value is inherited from the enclosing element, if not specified
**ISO-date**: indicates that an attribute value should be a legal ISO date in the form 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

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 SGML 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,[14] a date like “September 22, 1968” would be written “1968-09-22”. The parameter entity *ISO-date* expands to “CDATA”.

The relevant portion of the DTD looks like this:

```
<!-- 3.8.3:  TEI Keywords                                    -->

<!-- These parameter entities are used as keywords to        -->
<!-- express rules or constraints which cannot be fully      -->
<!-- expressed in SGML; their expansions show the nearest    -->
<!-- SGML equivalent.                                        -->


<!ENTITY % INHERITED '#IMPLIED' >
<!ENTITY % ISO-date 'CDATA' >
<!ENTITY % extPtr 'CDATA' >
<!-- This fragment is used in sec. 3.6.1                     -->
```

standard attribute values, and parameter entity names are less easily modified.

[14] Defined by ISO 8601: 1988, *Data elements and interchange formats — Information interchange — Representation of dates and times* ([Geneva]: International Organization for Standardization, 1988).

## Part 7

### *Alphabetic Reference List of Tags*

Class Name: **agent** (i.e. agent (individual or corporate body))
Description: defines a group of elements which contain names of individuals or corporate bodies.
Attributes: [None: global and inherited attributes only.]
Remarks: This class is used in the **<resp>** element, to allow a statement of responsibility to apply to an individual or a body.
Part: type=base name=core>.
Member of classes:
DTD file:
Discussed in 6.4, “Names, Numbers, Dates, and Abbreviations,” [in separate fascicle].

Class Name: **analysis**
Description: defines additional analytic attributes which are enabled for all elements when the additional tag set for analysis is selected.
Attributes: [None: global and inherited attributes only.]
Part: type=top name=ling>.
Member of classes:
DTD file:
Discussed in 16, “Segmentation and Alignment,” [in separate fascicle].

Class Name: **bibl**
Description: bibliographic elements.
Attributes: [None: global and inherited attributes only.]
Remarks:
Part: type=base name=core>.
Member of classes: inter common
DTD file:
Discussed in 6.10, “Bibliographic Citations,” [in separate fascicle].

Class Name: **bibl.part** (i.e. bibliographic citation part)

Description: elements which can appear within bibliographic citations, but cannot appear freely in running text outside bibliographic citations.
Attributes: [None: global and inherited attributes only.]
Remarks: This class is used in defining the content model of **<bibl>**.
Part: type=base name=core>.
Member of classes:
DTD file: xmisc
Discussed in 6.10, “Bibliographic Citations,” [in separate fascicle].

Class Name: **certainty**
Description: defines additional attributes which are enabled for all elements when the additional tag set for certainty and uncertainty is selected.
Attributes: [None: global and inherited attributes only.]
Part: type=top name=ce>.
Member of classes:
DTD file:
Discussed in 19, “Certainty,” [in separate fascicle].

Class Name: **chunk**
Description: includes all elements which can occur between, but not within, paragraphs and other chunks.
Attributes: [None: global and inherited attributes only.]
Remarks: This element class does not contain all those elements which can appear between chunks: the class *inter* contains a set of elements which can appear either within or between chunks. Unlike elements of that class, chunks cannot occur within chunks. In prose, this means the elements in this class can appear between but not within paragraphs.
Part: type=base name=core>.
Member of classes: common
DTD file:
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **chunk.seq** (i.e. chunk sequence)
Description: defines a sequence of *chunks*, i.e. of paragraph-level elements of the class *chunk* or of the class *inter*.
Remarks:
Expansion: '(%m.inter | %m.chunk;)*'
Discussed in 3.7, “Element Classes,” on page 22.

Class Name: **common**

Description: common chunk- and inter-level elements.

Attributes: [None: global and inherited attributes only.]

Remarks: This class defines the set of chunk- and inter-level elements available in all bases; it is used in defining the standard models *chunk.seq* and *specialPara* in the general and mixed bases.

Part: type=base name=core>.

Member of classes:

DTD file:

Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **component** (i.e. component for mixed or general base)

Description: defines the set of component-level elements for use with the mixed or general base; these are elements which can appear directly within text bodies or text divisions.

Expansion: '(%m.common %mix.verse %mix.drama %mix.spoken %mix.letters %mix.dictionaries %mix.terminology'

Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **component** (i.e. component for dictionary)

Description: defines the set of component-level elements for dictionary are elements which can appear directly within text bodies or text these divisions.

Expansion: '(%m.common | %m.comp.dictionaries)'

Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **component** (i.e. component for drama)

Description: defines the set of component-level elements for drama elements which can appear directly within text bodies or text divisions. these are

Expansion: '(%m.common | %m.comp.drama)'

Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **component** (i.e. component for letters)

Description: defines the set of component-level elements for letters elements which can appear directly within text bodies or text divisions. these are
Expansion: '(%m.common | %m.comp.letters)'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **component** (i.e. component for prose)
Description: defines the set of component-level elements for prose elements which can appear directly within text bodies or text divisions. these are
Expansion: '(%m.common)'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **component** (i.e. component for transcription of spoken texts)
Description: defines the set of component-level elements for spoken texts these are elements which can appear directly within text bodies or text divisions.
Expansion: '(%m.common | %m.comp.spoken)'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **component** (i.e. component for terminology)
Description: defines the set of component-level elements for terminology are elements which can appear directly within text bodies or text these divisions.
Expansion: '(%m.common | %m.comp.terminology)'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **component** (i.e. component for verse)
Description: defines the set of component-level elements for verse elements which can appear directly within text bodies or text divisions. these are
Expansion: '(%m.common | %m.comp.verse)'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **component.plus** (i.e. component sequence)
Description: defines a sequence of components as needed in the general base tag set, allowing components from any base to be used, but preventing their mixing.
Remarks: When used in a content model, this entity requires at least one component-level element to occur.
Expansion: '(%gen.verse %gen.drama %gen.spoken %gen.letters %gen.dictionary %gen.terminology TEI...end ) | ( (%m.common)+, ( %gen.verse %gen.drama %gen.spoken %gen.letters %gen.dictionary %gen.terminology TEI...end )?'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **component.seq** (i.e. component sequence)
Description: defines a sequence of components as needed in the general base tag set, allowing components from any base to be used, but preventing their mixing.
Remarks: When used in a content model, this entity, like other entities with names of the form “x.seq”, allows zero or more occurrences of its content to occur.
Expansion: '(%m.common)*, ( %gen.verse %gen.drama %gen.spoken %gen.letters %gen.dictionary %gen.terminology TEI...end )?'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **component.seq** (i.e. component-sequence)
Description: defines a sequence of component-level elements (such as paragraphs or lists) which can occur directly within text divisions and in similar positions.
Remarks: This parameter entity is used in each base tag set to define the content of **<div>** and similar elements.
Expansion: '(%component )*'
Discussed in 3.7, “Element Classes,” on page 22.

Class Name: **data**
Description: phrase-level elements containing names, dates, numbers, measures, and similar data.
Attributes: [None: global and inherited attributes only.]
Remarks:
Part: type=base name=core>.

Member of classes: phrase
DTD file:
Discussed in 6.4, “Names, Numbers, Dates, and Abbreviations,” [in separate fascicle].

Class Name: **declarable**

Description: elements which may be independently selected (using the special purpose *decls* attribute) from a candidate list of declarations within a TEI header.
Attributes:

*default*: indicates whether or not this element is selected by default when its parent is selected.
Data type: (YES|NO)

YES : This element is selected if its parent is selected
NO : This element can only be selected explicitly, unless it is the only one of its kind, in which case it is selected if its parent is selected.

Default value: NO
This attribute is mandatory when applicable.

Remarks: The rules governing the association of declarable elements with individual parts of a TEI text are fully defined in chapter 26.3, “Associating Contextual Information with a Text,” [in separate fascicle]. Only one element of a particular type may carry the value *default=yes*.
Part: type=aux name=hdr>.
Member of classes:
DTD file:
Discussed in 26.3, “Associating Contextual Information with a Text,” [in separate fascicle].

Class Name: **declaring**

Description: groups elements which may be independently associated with a particular declarable element within the header, thus overriding the inherited default for that element.
Attributes:

*decls*: identifies one or more *declarable elements* within the header, which are understood to apply to the element bearing this attribute and its content.
Data type: IDREFS
Value: must identify a set of declarable elements of different types.
Default value: #IMPLIED
This attribute is mandatory when applicable.

Remarks: The rules governing the association of declarable elements with individual parts of a TEI text are fully defined in chapter 26.3, “Associating Contextual Information with a Text,” [in separate fascicle].
Part: type=aux name=hdr>.
Member of classes:
DTD file:
Discussed in 26.3, “Associating Contextual Information with a Text,” [in separate fascicle].

Class Name: **demographic**
Description: elements describing demographic characteristics of the participants in a linguistic interaction.
Attributes: [None: global and inherited attributes only.]
Remarks:
Part: type=aux name=hdr>.
Member of classes:
DTD file:
Discussed in 26.2.2, “The Participants Description,” [in separate fascicle].

Class Name: **divbot** (i.e. Bottom-of-division elements)
Description: groups elements which can occur at the end of a text division; for example, trailer, byline, etc.
Attributes: [None: global and inherited attributes only.]
Part: type=base name=core>.
Member of classes:
DTD file: xmisc
Discussed in 7.2, “Elements Common to All Divisions,” [in separate fascicle].

Class Name: **divn**
Description: structural elements which behave in the same way as divisions.
Attributes:

*type*: specifies a name conventionally used for this level of subdivision, e.g. “act”, “volume”, “book”, “section”, “canto”, etc.
Data type: CDATA
Value: any string of characters
Default value: #CURRENT
This attribute is recommended.

*org*: specifies how the content of the division is organized.
Data type: (composite | uniform)

Legal values are:
composite : composite content: i.e. no claim is made about the sequence in which the immediate contents of this division are to be processed, or their inter-relationships.
uniform : uniform content: i.e. the immediate contents of this element are regarded as forming a logical unit, to be processed in sequence.
Default value: uniform
This attribute is optional.

*complete*: specifies whether or not this division is complete with respect to the original source.
Data type: (YES | NO)
Legal values are:
YES : the full text of the original has been transcribed.
NO : a sample of the original text has been transcribed.
Default value: YES
This attribute is optional.

*part* (i.e. part): indicates from which part of the original source the division has been extracted in the case of a sampled division.
Data type: (initial | medial | final | unknown | complete)
Legal values are:
initial : division lacks material at end.
medial : division lacks material at start and end.
final : division lacks material at start.
unknown : position of sampled material within original unknown.
complete : division is not a sample.
Default value: #IMPLIED
Example:

Remarks:
This attribute is optional.

Part: type=base name=core>.
Member of classes:
DTD file:
Discussed in 7, "Default Text Structure," [in separate fascicle].

Class Name: **divtop** (i.e. top-of-div elements)
Description: groups elements which can occur at the start of a text block: head, epigraph, byline, etc.
Attributes: [None: global and inherited attributes only.]
Part: type=base name=core>.
Member of classes:
DTD file: xmisc
Discussed in 7.2, "Elements Common to All Divisions," [in separate fascicle].

Class Name: **edit**
Description: phrase-level elements for simple editorial correction and transcription.
Attributes: [None: global and inherited attributes only.]
Remarks:
Part: type=base name=core>.
Member of classes: phrase
DTD file:
Discussed in 6.5, “Simple Editorial Changes,” [in separate fascicle].

Entity Name: **extPtr** (i.e. extended-pointer expression)
Description: used as the declared value of an attribute, indicates that all values of that attribute must be valid expressions in the TEI extended pointer notation defined in section 16.2, “External pointers and references,” [in separate fascicle].
Remarks:
Expansion: 'CDATA'
Discussed in 16.2, “External pointers and references,” [in separate fascicle].

Class Name: **te.global** (i.e. global attributes for terminological data.)
Description: defines attributes for all elements in documents which use the base tag set for terminological data.
Attributes:

*group*: indicates the group (term and related elements) to which this element should be associated.
Data type: CDATA
Value: any string matching the *n* attribute value on the **<term>** element to which the group is attached.
Default value: #IMPLIED
Example:

Remarks: The *group* attribute provides a specialized pointing mechanism for use within **<termEntry>** elements.
This attribute is optional.

*depend*: indicates the parent element to which this element should be associated.
Data type: CDATA
Value: any string matching the *n* attribute value on the element to which the dependent element is attached.
Default value: #IMPLIED

Example:

Remarks: If no *depend* or *group* attribute is specified, the element is assumed to relate to the most recently specified **<term>** or **<otherForm>** element.
This attribute is optional.

Remarks: The attributes shared by this element class are used for linking elements, possibly not adjacent in the record, which are related (e.g. a grammatical annotation and the term it describes). The usual ID/IDREF-based pointing mechanisms are not used, primarily because the requirement that SGML ID values be unique within a document places an impractical burden on the encoder when terminological data from several sources are to be merged.
Part: type=base name=term>.
Member of classes:
DTD file:
Discussed in 13.3.2, “Flat Term Entries Using Rules of Adjacency,” [in separate fascicle].

Class Name: **fs** (i.e. feature-structures)
Description: defines additional attributes for all elements, which are enabled when the additional tag set for feature-structure notation is selected.
Attributes: [None: global and inherited attributes only.]
Remarks: [No attributes declared yet.]
Part: type=top name=ling>.
Member of classes:
DTD file:
Discussed in 18, “Feature Structure Analysis,” [in separate fascicle].

Entity Name: **gen.dictionary** (i.e. dictionary part of general-base component sequence)
Description: contains a string used in constructing the definition of component sequence used in the general base tag set.
Expansion: '((%m.comp.dictionary), (%m.common | %m.comp.dictionary)*) |'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **gen.dictionary** (i.e. dictionary part of general-base component sequence)
Description: contains a string used in constructing the definition of component sequence used in the general base tag set.
Remarks: This version of this entity is used if the dictionary base is not selected.
Expansion: ''
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **gen.drama** (i.e. drama part of general-base component sequence)
Description: contains a string used in constructing the definition of component sequence used in the general base tag set.
Expansion: '((%m.comp.drama), (%m.common | %m.comp.drama)*) |'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **gen.drama** (i.e. drama part of general-base component sequence)
Description: contains a string used in constructing the definition of component sequence used in the general base tag set.
Remarks: This version of this entity is used if the drama base is not selected.
Expansion: ''
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **gen.letters** (i.e. letters part of general-base component sequence)
Description: contains a string used in constructing the definition of component sequence used in the general base tag set.
Expansion: '((%m.comp.letters), (%m.common | %m.comp.letters)*) |'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **gen.letters** (i.e. letters part of general-base component sequence)
Description: contains a string used in constructing the definition of component sequence used in the general base tag set.
Remarks: This version of this entity is used if the base tag set for letters and memoranda is not selected.
Expansion: ''

Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **gen.spoken** (i.e. spoken-text part of general-base component sequence)
Description: contains a string used in constructing the definition of component sequence used in the general base tag set.
Expansion: '((%m.comp.spoken), (%m.common | %m.comp.spoken)*) |'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **gen.spoken** (i.e. spoken-text part of general-base component sequence)
Description: contains a string used in constructing the definition of component sequence used in the general base tag set.
Remarks: This version of this entity is used if the spoken base is not selected.
Expansion: ''
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **gen.terminology** (i.e. terminology part of general-base component sequence)
Description: contains a string used in constructing the definition of component sequence used in the general base tag set.
Expansion: '((%m.comp.terminology), (%m.common | %m.comp.terminology)*) |'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **gen.terminology** (i.e. terminology part of general-base component sequence)
Description: contains a string used in constructing the definition of component sequence used in the general base tag set.
Remarks: This version of this entity is used if the terminology base is not selected.
Expansion: ''
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **gen.verse** (i.e. verse part of general-base component sequence)
Description: contains a string used in constructing the definition of component sequence used in the general base tag set.
Expansion: '((%m.comp.verse), (%m.common | %m.comp.verse)*) |'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **gen.verse** (i.e. verse part of general-base component sequence)
Description: contains a string used in constructing the definition of component sequence used in the general base tag set.
Remarks: This version of this entity is used if the verse base is not selected.
Expansion: ''
Discussed in 3.7, “Element Classes,” on page 22.

Class Name: **global**
Description: defines attributes common to all elements in the TEI encoding scheme.
Attributes:

*id* (i.e. identifier): provides a unique identifier for the element bearing the ID value.
Data type: ID
Value: any valid SGML name (in the reference concrete syntax, a name must begin with a letter and contain only letters, numeric digits, hyphen, and full stop).
Default value: #IMPLIED
Example:

```
<p id=names>Paragraph with the ID <mentioned>names</>....
<p id=dates>Paragraph with the ID <mentioned>dates</>....
```

Remarks: The *id* attribute may be used to specify a canonical reference for an element; see section 6.9, “Reference Systems,” [in separate fascicle].
This attribute is optional.

*n* (i.e. number, name, etc.): gives a number (or other label) for an element, which is not necessarily unique within the document.
Data type: CDATA
Value: any string of characters; often, but not necessarily, numeric.
Default value: #IMPLIED
Example:

Remarks: The *n* attribute may be used to specify the numbering of chapters, sections, list items, etc.; it may also be used in the specification of a standard reference system for the text.
This attribute is optional.

*lang* (i.e. language): indicates the language of the element content, usually using a two- or three-letter code from ISO 639.
Data type: IDREF
Value: The value must be the identifier specified for a writing system declaration declared in the TEI header, as described in section 5, “The TEI Header,” [in separate fascicle].
Default value: %INHERITED
Example:

```
<p lang=EN>The only surviving work by <person>Ari</> (died 1148)
is the ten-page <title lang=IS>&Iacute;slendingab&oacute;k</>
(<title lang=LA>Libellus Islandorum</>), written in the early
twelfth century.
```

Remarks: If no value is specified for *lang*, the *lang* value for the immediately enclosing element is inherited; for this reason, a value should always be specified on the outermost element (**<TEI.2>**).
This attribute is optional.

*rend* (i.e. rendition or presentation): indicates how the element in question was rendered or presented in the source text.
Data type: CDATA
Value: any string of characters; if the typographic rendition of a text is to be systematically recorded, a systematic set of values for the *rend* attribute should be defined.
Default value: #IMPLIED
Example:

Remarks: These Guidelines make no binding recommendations for the values of the *rend* attribute; the characteristics of visual presentation vary too much from text to text and the decision to record or ignore individual characteristics varies too much from project to project. Some potentially useful conventions are noted from time to time at appropriate points in the Guidelines.
This attribute is optional.

Remarks: The global attributes described here are made part of the attribute definition list declaration of each element by including the string “%a.global” in each such declaration. Some global

attributes are made available when certain base or additional tag sets are selected; these are incorporated into the global attributes by references to the appropriate parameter entities. When the tag sets in question have

not been selected, the parameter entities in question expand to the empty string.
Part: type=base name=core>.
Member of classes: clterm cllink clana clfs clcert clspoken
DTD file:
Discussed in 3.5, “Global Attributes,” on page 10.

Class Name: **hqinter**
Description: intermediate-level elements related to highlighting.
Attributes: [None: global and inherited attributes only.]
Remarks:
Part: type=base name=core>.
Member of classes: inter common
DTD file:
Discussed in 6.3, “Highlighting and Quotation Marks,” [in separate fascicle].

Class Name: **hqphrase**
Description: phrase-level elements related to highlighting.
Attributes: [None: global and inherited attributes only.]
Remarks:
Part: type=base name=core>.
Member of classes: phrase
DTD file:
Discussed in 6.3, “Highlighting and Quotation Marks,” [in separate fascicle].

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

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

The most prominent example of attribute value inheritance is the TEI global attribute *lang*.

Expansion: '#IMPLIED'

Discussed in 3.7, “Element Classes,” on page 22.

Class Name: # inter

Description: elements of the intermediate (inter-level) class, which can occur both within paragraphs and between paragraphs.

Attributes: [None: global and inherited attributes only.]

Remarks: This element class contains a subset of those elements which can appear in the unstructured “soup” with which paragraph and other elements at the lowest level of crystal structures are filled: specifically all the elements which can also occur as structural elements in their own right. In prose, this means the elements in this class can appear both within and between paragraphs. This class is thus distinct from the purely phrase-level elements which can appear only within soup, and not on their own; the latter class, in keeping with this metaphor, is called “broth”; it is represented by the class *phrase*. Cf. also the class *chunks*.

Part: type=base name=core>.

Member of classes:

DTD file:

Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: # ISO-date

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

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

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

Expansion: 'CDATA'

Discussed in 3.8.3, “Parameter Entities for TEI Keywords,” on page 39.

Entity Name: **mix.dictionary** (i.e. mixed-base dictionary components)
Description: contains a string used in constructing the definition of *component* used in the mixed base tag set.
Expansion: '| %m.comp.dictionary'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **mix.drama** (i.e. mixed-base drama components)
Description: contains a string used in constructing the definition of *component* used in the mixed base tag set.
Expansion: '| %m.comp.drama'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **mix.letters** (i.e. mixed-base components for letters and memoranda)
Description: contains a string used in constructing the definition of *component* used in the mixed base tag set.
Expansion: '| %m.comp.letters'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **mix.spoken** (i.e. mixed-base spoken-text components)
Description: contains a string used in constructing the definition of *component* used in the mixed base tag set.
Expansion: '| %m.comp.spoken'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **mix.terminology** (i.e. mixed-base terminology components)
Description: contains a string used in constructing the definition of *component* used in the mixed base tag set.
Expansion: '| %m.comp.terminology'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **mix.verse** (i.e. mixed-base verse components)
Description: contains a string used in constructing the definition of *component* used in the mixed base tag set.
Expansion: '| %m.comp.verse'
Discussed in 3.7, “Element Classes,” on page 22.

Class Name: **linking**
Description: defines additional attributes for hypertext and other linking, which are enabled for all elements when the additional tag set for linking is selected.
Attributes:

*corresp* (i.e. correspondents): specifies the identifiers of corresponding elements.
Data type: IDREFS
Value: one or more valid SGML identifiers for elements in the current document separated by white space.
Default value: #IMPLIED
Example:

Remarks:
This attribute is optional.

*link* (i.e. link): specifies the identifiers of elements that may be linked with this one to form an aggregate.
Data type: IDREFS
Value: one or more valid SGML identifiers for elements in the current document separated by white space.
Default value: #IMPLIED
Example:

Remarks:
This attribute is optional.

*partof*: specifies the identifier of a **<join>** element in the current document representing an aggregate of which this element is a component part.
Data type: IDREF
Value: a valid SGML identifier attached to a **<join>** element.
Default value: #IMPLIED
Example:

Remarks:
This attribute is optional.

*prev* (i.e. previous): specifies the identifier of an element which may be linked with the current element to form an aggregate, and which immediately precedes it in the aggregate.
Data type: IDREF
Value: a valid SGML identifier for an element in the current document.
Default value: #IMPLIED
Example:

Remarks:
This attribute is optional.

*next* (i.e. next): specifies the identifier of an element which may be linked with the current element to form an aggregate, and which immediately follows it in the aggregate.
Data type: IDREF
Value: a valid SGML identifier for an element in the current document.
Default value: #IMPLIED
Example:

Remarks:
This attribute is optional.

Part: type=top name=ling>.
Member of classes:
DTD file:
Discussed in 16, “Segmentation and Alignment,” [in separate fascicle].

Class Name: **lists**
Description: list-like elements.
Attributes: [None: global and inherited attributes only.]
Remarks:
Part: type=base name=core>.
Member of classes: inter common
DTD file:
Discussed in 6.7, “Lists,” [in separate fascicle].

Class Name: **loc**
Description: elements used for purposes of location and reference
Attributes: [None: global and inherited attributes only.]
Remarks:
Part: type=base name=core>.
Member of classes: phrase
DTD file:

Discussed in 6.6, “Simple Cross References,” [in separate fascicle].

Class Name: **names**
Description: elements with proper nouns as content.
Attributes:

*key*: provides an alternative identifier for the object being named, such as a database record key.
Data type: CDATA
Value: any string
Default value: #IMPLIED
Example:

```
<name type='place'>Montaillou</name>
is not a large parish.
At the time of the events which led to <name type='person' key=BXII>
Fournier's</name> investigations,
the local population consisted of between 200 and
250 inhabitants.
```

Remarks: The value may be a unique identifier from a database, or simply a more explicit name for the referent. Its purpose is only to record an identification; if the analysis leading to the identification is to be recorded as well, the analytic tags described in chapter 18, “Feature Structure Analysis,” [in separate fascicle] should be used in addition or instead.
This attribute is optional.

*reg* (i.e. regularization): gives a normalized or regularized form of the proper name
Data type: CDATA
Value: Any string of characters.
Default value: #IMPLIED
Example:

```
At the time of the events which led to
<name type=person reg='Benedict XII, Pope of Avignon
(Jacques Fournier)'>Fournier's</name> investigations,
the local population consisted of between 200 and
250 inhabitants.
```

Remarks: In providing a “regularized” form, no claim is made that the form in the source text is incorrect; the regularized form is simply that chosen as the main form for purposes of unifying variant forms under a single heading.
This attribute is optional.

Remarks:
Part: type=base name=core>.

Member of classes:
DTD file: dummy
Discussed in 6.4.1, “Names,” [in separate fascicle].

Class Name: **notes**
Description: note-like elements.
Attributes: [None: global and inherited attributes only.]
Remarks:
Part: type=base name=core>.
Member of classes: inter common
DTD file:
Discussed in 6.8, “Notes,” [in separate fascicle].

Entity Name: **paraContent** (i.e. paragraph content)
Description: defines the legal version for paragraphs and similar elements.
Remarks: [Dummy version of entdoc.]
Expansion: '(#PCDATA | %m.phrase | %m.inter)*'
Discussed in 3.7, “Element Classes,” on page 22.

Class Name: **phrase**
Description: includes elements which can occur at the level of individual words or phrases.
Attributes: [None: global and inherited attributes only.]
Remarks: This class of elements can occur only within larger elements of the class *inter* or *chunk*. In prose, this means these elements can occur within paragraphs, list items, lines of verse, etc.
Part: type=base name=core>.
Member of classes:
DTD file: dummy
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **phrase**
Description: defines a phrase as character data or any phrase-level element.
Remarks: This entity is used in the declaration of *phrase.seq*.
Expansion: '(#PCDATA | %m.phrase)'
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **phrase.seq** (i.e. phrase sequence)
Description: defines a sequence of character data and phrase-level elements.
Expansion: '(%phrase )*'
Discussed in 7.7, "Overall Structure of the Prose DTD," [in separate fascicle].

Class Name: **pointer**
Description: elements which point to other elements, using IDREFs.

Attributes:

*type*: categorizes the pointer in some respect, using any convenient set of categories.
Data type: CDATA
Value: The type should indicate the intended function of the pointer, or the rhetorical relationship between its source and target.
Default value: #IMPLIED

This attribute is optional.

*resp* : specifies the creator of the pointer.
Data type: CDATA
Value: any string of characters, usually the initials or name of the creator.
Default value: #IMPLIED
This attribute is optional.

*crdate* : specifies when the pointer was created.
Data type: CDATA
Value: any string representing a date.
Default value: #IMPLIED
This attribute is optional.

*targType* : specifies the kinds of elements to which this pointer may point.

Data type: NAMES
Value: A list of names of SGML elements defined in the DTD of the current document.
Default value: #IMPLIED
Remarks: If this attribute is supplied, every element specified as a target must be of one or the other of the types specified. An application may choose whether or not to report failures to satisfy this constraint as errors, but may not access an element of the right identifier but the wrong type.

This attribute is optional.

*targOrder* : where more than one identifier is supplied as the value of the *target* attribute, this attribute specifies whether the order in which they are supplied is significant.

Data type: (Y | N | U)

Legal values are:

Y : Yes: the order in which IDREFs are specified as the value of a *target* attribute should be followed when combining the targeted elements.

N : No: the order in which IDREFs are specified as the value of a *target* attribute has no significance when combining the targeted elements.

U : Unspecified: the order in which IDREFs are specified as the value of a *target* attribute may or may not be significant.

Default value: U

This attribute is optional.

*evaluate* : specifies the intended meaning when the target of a pointer is itself a pointer.

Data type: (all | one | none)

all : if the element pointed to is itself a pointer, then the target of that pointer will be taken, and so on, until an element is found which is not a pointer.

one : if the element pointed to is itself a pointer, then its target (whether a pointer or not) is taken as the target of this pointer.

none : no further evaluation of targets is carried out beyond that needed to find the element specified in the pointer's target.

Default value: #IMPLIED

Remarks: If no value is given, the application program is responsible for deciding (possibly on the basis of user input) how far to trace a chain of pointers.

This attribute is optional.

Remarks:
Part: type=base name=core>.
Member of classes:
DTD file:
Discussed in 6.6, “Simple Cross References,” [in separate fascicle].

Class Name: **refsys** (i.e. reference system elements)
Description: milestone-style elements used in reference systems
Attributes: [None: global and inherited attributes only.]
Remarks:
Part: type=base name=core>.
Member of classes:
DTD file: xmisc
Discussed in 3.7, “Element Classes,” on page 22.

Class Name: **seg**
Description: elements for arbitrary segmentation.
Attributes: [None: global and inherited attributes only.]
Remarks:
Part: type=base name=core>.
Member of classes: phrase
DTD file:
Discussed in 6.11, “Segmentation,” [in separate fascicle].

Entity Name: **specialPara** (i.e. 'special' paragraph content)
Description: defines the content model of elements such as notes or list items, which either contain a series of component-level elements or else have the same structure as a paragraph, containing a series of phrase-level and inter-level elements.
Remarks: [Dummy.]
Expansion: '(((%m.chunk), (%component.seq)) | (%paraContent))'
Discussed in 3.7, “Element Classes,” on page 22.

Class Name: **spoken**
Description: defines additional attributes which are enabled for all elements when the base tag set for transcription of spoken texts is selected.
Attributes: [None: global and inherited attributes only.]
Part: type=top name=spoken>.
Member of classes:
DTD file:
Discussed in 10, “Base Tag Set for Transcriptions of Spoken Texts,” [in separate fascicle].

# <TEI...end> (i.e. TEI end-sequence (dummy element))

Description: an undefined element, whose generic identifier is used in the definition of mixed-base component sequences.
Attributes: [None: global and inherited attributes only.]
Example:

Remarks: There is no **<TEI...end>** element declared in any TEI document type declaration. The identifier is used in the declaration of the combined base tag sets, as a terminator to a sequence of expressions each followed by a '|'; the use of this dummy terminator simplifies the syntax of these expressions, which would otherwise be awkwardly complex. It is recommended that extensions to the TEI tag sets avoid use of this identifier.
Part: base tag set for common core features.
Member of classes: global
DTD file:
Data description: Imaginary element.
May occur within:
May contain: EMPTY
Element Declaration:

```
<!ELEMENT TEI...end     - O  EMPTY                              >
```

Attribute List Declaration:

```
<!ATTLIST TEI...end          %a.global;                         >
```

Discussed in 3.7.3, “The TEICLAS2.ENT File,” on page 26.
This tag is optional.

# <TEI.2> (i.e. TEI 2 document)

Description: contains a single TEI-conformant document, comprising a TEI header and a text, either in isolation or as part of a Teicorpus.
Attributes: [None: global and inherited attributes only.]
Example:

```
<tei.2>
  <teiHeader>
    <fileDesc>
       <titleStmt>
          <title>The shortest TEI Document Imaginable</title>
       <publicationStmt>
          <p>Published as part of TEI P2.
       </publicationStmt>
    <sourceDesc>
          <p>No source: this is an original work.
    </sourceDesc>
```

```
        </teiHeader>
        <text>
          <body>
           <p>This is about the shortest TEI document imaginable.
          </body>
        </text>
      </tei.2>
```

Remarks: This element is required.
Part: auxiliary tag set for common core features.
Member of classes:
DTD file:
Data description: Must contain one TEI header and one text.
May occur within:
May contain: text TeiHeader
Element Declaration:

```
<!ELEMENT TEI.2         - O  (TeiHeader, text)                 >
```

Attribute List Declaration:

```
<!ATTLIST TEI.2              %a.global;                        >
```

Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.
This tag is required.

Entity Name: **TEI.analysis** (i.e. TEI simple analytic mechanisms DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for simple analytic mechanisms.
Remarks: To include elements and attributes for simple analytic mechanisms in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.analysis.dtd** (i.e. TEI analysis-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for simple analysis.
Remarks: This entity is declared with this value when the user includes the base tag set for simple analysis.
Expansion: system 'teiana2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.analysis.ent** (i.e. TEI analysis-base entities)
Description: identifies the file containing parameter entity declarations for the element classes defined in the base tag set for simple analysis.
Remarks: This entity is declared with this value when the user includes the base tag set for simple analysis.
Expansion: system 'teiana2.ent'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.back.dtd**
Description: defines the file in which tags for back matter are defined.
Remarks: This parameter entity is used in each base tag set to include the back-matter tags, which are common to all bases.
Expansion: system 'TEIback2.dtd'
Discussed in 7.5, “Back Matter,” [in separate fascicle].

Entity Name: **TEI.certainty** (i.e. TEI certainty DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for certainty.
Remarks: To include elements and attributes for certainty in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.5, “Global Attributes,” on page 10.
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.certainty.dtd** (i.e. TEI certainty-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for certainty and uncertainty.
Remarks: This entity is declared with this value when the user includes the base tag set for certainty and uncertainty.
Expansion: system 'teicert2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.certainty.ent** (i.e. TEI certainty-base entities)
Description: identifies the file containing parameter entity declarations for the element classes defined in the base tag set for certainty and uncertainty.
Remarks: This entity is declared with this value when the user includes the base tag set for certainty and uncertainty.
Expansion: system 'teicert2.ent'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.core.dtd** (i.e. TEI core)
Description: identifies the file containing element and attribute list declarations for the TEI core elements.
Remarks: This entity is included in all TEI DTDs.
Expansion: system 'TEIcore2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.corpus** (i.e. TEI corpora DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for the description of corpora.
Remarks: To include elements and attributes for corpora in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.5, “Global Attributes,” on page 10.
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

## <TEI.corpus.2> (i.e. TEI corpus)

Description: contains the whole of a TEI encoded corpus, comprising a single corpus header and one or more TEI.2 elements, each containing a single text header and a text.
Attributes: [None: global and inherited attributes only.]
Example:

```
<TEI.corpus.2>
    <TeiHeader> ...
    <!-- header for corpus-level information -->
    </TeiHeader>
    <TEI.2 id=T1>
         <TeiHeader> ...
         <!-- header for text-level information -->
```

```
          </TeiHeader>
          <text> ... </text>
     </TEI.2>
     <TEI.2 id=T2>
          <TeiHeader> ... </TeiHeader>
          <text> ... </text>
     </TEI.2>
     <!-- etc. -->
</TEI.corpus.2>
```

Remarks: This element is mandatory when applicable.
Part: auxiliary tag set for common core features.
Member of classes:
DTD file:
Data description: Must contain one TEI header for the corpus, and a series of **<TEI.2>** elements, one for each text.
May occur within:
May contain: tei.2 TeiHeader
Element Declaration:

```
<!ELEMENT TEI.corpus.2  - O  (TeiHeader, tei.2+)               >
```

Attribute List Declaration:

```
<!ATTLIST TEI.corpus.2       %a.global;                        >
```

Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.
Discussed in 26.1, “Varieties of Composite Text,” [in separate fascicle].
This tag is mandatory when applicable.

Entity Name: **TEI.corpus.dtd** (i.e. TEI corpus-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for corpora.
Remarks: This entity is declared with this value when the user includes the base tag set for corpora.
Expansion: system 'teicorp2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.dictionaries** (i.e. TEI dictionaries DTD fragment)

Description: controls the inclusion, in the DTD, of element and attribute declarations for dictionaries.
Remarks: To include elements and attributes for dictionaries in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.5, “Global Attributes,” on page 10.
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: # TEI.dictionaries.dtd (i.e. TEI dictionary-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for dictionaries.
Remarks: This entity is declared with this value when the user includes the base tag set for dictionaries.
Expansion: system 'teidict2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: # TEI.dictionaries.ent (i.e. TEI dictionary-base entities)
Description: identifies the file containing parameter entity declarations for the element classes defined in the base tag set for dictionaries.
Remarks: This entity is declared with this value when the user includes the base tag set for dictionaries.
Expansion: system 'teidict2.ent'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: # TEI.drama (i.e. TEI drama DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for drama.
Remarks: To include elements and attributes for drama in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.5, “Global Attributes,” on page 10.
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.drama.dtd** (i.e. TEI drama-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for drama.
Remarks: This entity is declared with this value when the user includes the base tag set for drama.
Expansion: system 'teidram2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.drama.ent** (i.e. TEI drama-base entities)
Description: identifies the file containing parameter entity declarations for the element classes defined in the base tag set for drama.
Remarks: This entity is declared with this value when the user includes the base tag set for drama.
Expansion: system 'teidram2.ent'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.elementClasses**
Description: file containing the parameter entity declarations which define element classes for content models, attributes shared among elements of a class, and global attributes.
Remarks:
Expansion: system 'teiclas2.ent'
Discussed in 3.6, “The TEI2.DTD File,” on page 14.
Discussed in 3.7, “Element Classes,” on page 22.

Entity Name: **TEI.elementNames**
Description: file containing parameter entity declarations for all generic identifiers of the encoding scheme.

Remarks: The parameter entities in this file all take the same form as the two shown below:

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

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

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

This allows generic identifiers to be renamed conveniently, to provide names in languages other than English, or to provide shorter names than those documented here. See further chapter 32, “Modifying TEI DTDs,” [in separate fascicle].

Expansion: system 'teigis.ent'
Discussed in 3.8.2, “Parameter Entities for Element Generic Identifiers,” on page 38.

Entity Name: **TEI.extensions.ent**
Description: file (if any) containing local modifications to the TEI element classes.
Remarks: This entity is embedded in the TEI DTDs before the TEI element classes are embedded. By default, the entity expands to the empty string; the user can override this default by declaring the entity with an appropriate value, typically this will take the form

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

Expansion: ''
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.extensions.dtd**
Description: file (if any) containing local modifications to the TEI DTDs.
Remarks: This entity is embedded in the TEI DTDs after the TEI element classes are embedded, and immediately before the base tag set and additional tag sets selected by the user are embedded. By default, the entity expands to the empty string; the user can override this default by declaring the entity with an appropriate value, typically this will take the form

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

Expansion: ''
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.figures** (i.e. TEI graphics and figures DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for graphics and figures.
Remarks: To include elements and attributes for graphics and figures in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.5, “Global Attributes,” on page 10.
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.figures.dtd** (i.e. TEI graphics-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for graphics and figures.
Remarks: This entity is declared with this value when the user includes the base tag set for graphics and figures.
Expansion: system 'teifig2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.formulas** (i.e. TEI formulas and tables DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for formulas and tables.
Remarks: To include elements and attributes for formulas and tables in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.5, “Global Attributes,” on page 10.
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.formulas.dtd** (i.e. TEI formulas-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for formulas and tables.
Remarks: This entity is declared with this value when the user includes the base tag set for formulas and tables.
Expansion: system 'teiform2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.front.dtd**
Description: defines the file in which tags for front matter are defined.
Remarks: This parameter entity is used in each base tag set to include the front-matter tags, which are common to all bases.
Expansion: system 'TEIfron2.dtd'
Discussed in 7.3, “Front Matter,” [in separate fascicle].

Entity Name: **TEI.fs** (i.e. TEI feature structures DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for feature structures.
Remarks: To include elements and attributes for feature structures in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.fs.dtd** (i.e. TEI fs-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for feature structures.
Remarks: This entity is declared with this value when the user includes the base tag set for feature structures.
Expansion: system 'teifs2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.fs.ent** (i.e. TEI fs-base entities)
Description: identifies the file containing parameter entity declarations for the element classes defined in the base tag set for feature structures.
Remarks: This entity is declared with this value when the user includes the base tag set for feature structures.
Expansion: system 'teifs2.ent'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.general** (i.e. TEI general base DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for the “general” combined base.

Remarks: To include elements and attributes for the “general” base in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.general.dtd** (i.e. TEI general-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for the “general” base tag set.
Remarks: This entity is declared with this value when the user includes the “general” mixed-base tag set.
Expansion: system 'teigen2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.header** (i.e. TEI header)
Description: identifies the file containing element and attribute list declarations for the TEI header.
Remarks: This entity is included in all TEI DTDs.
Expansion: system 'teihdr2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.letters** (i.e. TEI letters and memoranda DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for letters and memoranda.
Remarks: To include elements and attributes for letters and memoranda in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.5, “Global Attributes,” on page 10.
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.letters.dtd** (i.e. TEI letters-base DTD)

Description: identifies the file containing element and attribute list declarations for the base tag set for letters and memoranda.
Remarks: This entity is declared with this value when the user includes the base tag set for letters and memoranda.
Expansion: system 'teilett2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.letters.ent** (i.e. TEI letters-base entities)
Description: identifies the file containing parameter entity declarations for the element classes defined in the base tag set for letters and memoranda.
Remarks: This entity is declared with this value when the user includes the base tag set for letters and memoranda.
Expansion: system 'teilett2.ent'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.linking** (i.e. TEI DTD fragment for linking tag set)
Description: controls the inclusion, in the DTD, of element and attribute declarations for linking, segmentation and alignment.
Remarks: To include elements and attributes for segmentation and alignment in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.linking.dtd** (i.e. TEI segmentation and alignment elements)
Description: identifies the file containing element and attribute list declarations for the additional tag set for segmentation and alignment.
Remarks: This entity is declared with this value when the user includes the additional tag set for segmentation and alignment.
Expansion: system 'teilink2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.linking.ent** (i.e. TEI linking-tag-set entities)
Description: identifies the file containing parameter entity declarations for the element classes defined in the additional tag set for segmentation and alignment.
Remarks: This entity is declared with this value when the user includes the additional tag set for segmentation and alignment.
Expansion: system 'teilink2.ent'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.mixed** (i.e. TEI ‘mixed' base DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for the “mixed” combined base.
Remarks: To include elements and attributes for the “mixed” base in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.mixed.dtd** (i.e. TEI mixed-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for the “mixed” base tag set.
Remarks: This entity is declared with this value when the user includes the base tag set for the “mixed” base.
Expansion: system 'teimix2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.names.dates** (i.e. TEI names and dates DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for detailed analysis of names and dates.
Remarks: To include elements and attributes for names and dates in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.5, “Global Attributes,” on page 10.
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.names.dates.dtd** (i.e. TEI names.dates-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for detailed analysis of names and dates.
Remarks: This entity is declared with this value when the user includes the base tag set for names and dates.
Expansion: system 'teind2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.nets** (i.e. TEI graph theoretic DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for graph theory (graphs, digraphs and other networks)
Remarks: To include elements and attributes for graphs and digraphs in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.5, “Global Attributes,” on page 10.
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.nets.dtd** (i.e. TEI networks-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for graph theory (graphs, digraphs and other networks).
Remarks: This entity is declared with this value when the user includes the base tag set for graph theory.
Expansion: system 'teinet2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.physical** (i.e. TEI physical description DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for physical description of the source text.
Remarks: To include elements and attributes for physical description in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.5, “Global Attributes,” on page 10.
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.physical.dtd** (i.e. TEI physical-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for physical description of the source text.
Remarks: This entity is declared with this value when the user includes the base tag set for physical description.
Expansion: system 'teiphys2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.prose** (i.e. TEI prose DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for prose.
Remarks: To include elements and attributes for prose in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.prose.dtd** (i.e. TEI prose-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for prose.
Remarks: This entity is declared with this value when the user includes the base tag set for prose.
Expansion: system 'teipros2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.prose.ent** (i.e. TEI prose-tag-set entities)
Description: identifies the file containing parameter entity declarations for the element classes defined in the base tag set for prose.
Remarks: This entity is declared with this value when the user includes the base tag set for prose.
Expansion: system 'TEIpros2.ent'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.spoken** (i.e. TEI spoken texts DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for spoken texts.
Remarks: To include elements and attributes for spoken texts in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.5, “Global Attributes,” on page 10.
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.spoken.dtd** (i.e. TEI spoken-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for spoken texts.
Remarks: This entity is declared with this value when the user includes the base tag set for spoken texts.
Expansion: system 'teispok2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.spoken.ent** (i.e. TEI spoken-base entities)
Description: identifies the file containing parameter entity declarations for the element classes defined in the base tag set for spoken texts.
Remarks: This entity is declared with this value when the user includes the base tag set for spoken texts.
Expansion: system 'teispok2.ent'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.structure.dtd**
Description: defines the file in which the default text structure used by many base tag sets is defined.
Remarks:
Expansion: system 'teistr2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.terminology** (i.e. TEI terminological data DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for terminological data.
Remarks: To include elements and attributes for terminological data in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.terminology.dtd** (i.e. TEI terminology-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for terminological data.
Remarks: This entity is declared with this value when the user includes the base tag set for terminological data.
Expansion: system 'teiterm2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.terminology.ent** (i.e. TEI terminology-base entities)
Description: identifies the file containing parameter entity declarations for the element classes defined in the base tag set for terminological data.
Remarks: This entity is declared with this value when the user includes the base tag set for terminological data.
Expansion: system 'teiterm2.ent'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.textcrit** (i.e. TEI text criticism DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for text criticism.
Remarks: To include elements and attributes for text criticism in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.textcrit.dtd** (i.e. TEI text criticism base DTD)
Description: identifies the file containing element and attribute list declarations for the additional tag set for text criticism.
Remarks: This entity is declared with this value when the user includes the additional tag set for text criticism.
Expansion: system 'teitc2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.textcrit.ent** (i.e. TEI text criticism entities)
Description: identifies the file containing parameter entity declarations for the element classes defined in the additional tag set for text criticism.
Remarks: This entity is declared with this value when the user includes the additional tag set for text criticism.
Expansion: system 'TEItc2.ent'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.verse** (i.e. TEI verse DTD fragment)
Description: controls the inclusion, in the DTD, of element and attribute declarations for verse.
Remarks: To include elements and attributes for verse in the DTD, the user should declare this entity with a value of “INCLUDE”; this will override the default.
Expansion: 'IGNORE'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.
Discussed in 3.6, “The TEI2.DTD File,” on page 14.

Entity Name: **TEI.verse.dtd** (i.e. TEI verse-base DTD)
Description: identifies the file containing element and attribute list declarations for the base tag set for verse.
Remarks: This entity is declared with this value when the user includes the base tag set for verse.
Expansion: system 'teivers2.dtd'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Entity Name: **TEI.verse.ent** (i.e. TEI verse-base entities)
Description: identifies the file containing parameter entity declarations for the element classes defined in the base tag set for verse.
Remarks: This entity is declared with this value when the user includes the base tag set for verse.
Expansion: system 'teivers2.ent'
Discussed in 3.2, “Core, Base, and Additional Tag Sets,” on page 3.

Class Name: **teiform** (i.e. TEI name form)
Description: defines an attribute (*TEIform*) common to all tags in the TEI scheme, and recommended for all user-defined extensions.
Attributes:

*TEIform* (i.e. TEI form of generic identifier): indicates the standard TEI name (generic identifier) for a given element.
Data type: NAME
Value: must be a valid SGML name; the default is specified in
Default value: #IMPLIED
Example:

```
<fn TEIform=note>This is a footnote; its tag uses a non-standard
 name defined by the user; the attribute TEIform indicates that
 the normal TEI name for the element is NOTE.</fn>
```

Remarks: In the TEI DTDs, the default value for this attribute is always the same as the generic identifier of the element. If an element is renamed using the techniques described in chapter 32, “Modifying TEI DTDs,” [in separate fascicle], the attribute declaration for *TEIform* will be left undisturbed; the default value will thus still be the standard TEI name for the element. TEI-aware application programs can thus process TEI-conformant documents which rename TEI elements, since by consulting the *TEIform* attribute value the application can learn the standard name for the element and process it accordingly.

In the normal course of events, this attribute will never be specified in a TEI-conformant document; all occurrences will have the default value. In some special circumstances, it can be useful to specify a non-default value on some instances of an element; this allows application programs to process correctly a locally defined element which usually corresponds to one TEI element (which would be expressed by the default value) but sometimes to another TEI element (which would be expressed by explicit values attached to the element instance).

This attribute is mandatory when applicable.

Remarks: The attribute *TEIform*, though common to all tags in the TEI encoding scheme, is not defined as part of the *global* class for technical reasons. Since its default value must be specified separately for each element type, no elements actually inherit the attribute from the element class *TEIform*; each defines the attribute separately.

Part: type=base name=core>.
Member of classes:
DTD file:
Discussed in 3.5, “Global Attributes,” on page 10.

Class Name: **terminology** (i.e. global attributes for terminological data.)
Description: defines attributes for all elements in documents which use the base tag set for terminological data.
Attributes:

*group*: indicates the group (term and related elements) to which this element should be associated.
Data type: CDATA
Value: any string matching the *n* attribute value on the **<term>** element to which the group is attached.
Default value: #IMPLIED
Example:

Remarks: The *group* attribute provides a specialized pointing mechanism for use within **<termEntry>** elements.
This attribute is optional.

*depend*: indicates the parent element to which this element should be associated.
Data type: CDATA
Value: any string matching the *n* attribute value on the element to which the dependent element is attached.
Default value: #IMPLIED
Example:

Remarks: If no *depend* or *group* attribute is specified, the element is assumed to relate to the most recently specified **<term>** or **<otherForm>** element.
This attribute is optional.

Remarks: The attributes shared by this element class are used for linking elements, possibly not adjacent in the record, which are related (e.g. a grammatical annotation and the term it describes). The usual ID/IDREF-based pointing mechanisms are not used, primarily because the requirement that SGML ID values be unique within a document places an impractical burden on the encoder when terminological data from several sources are to be merged.

Part: type=base name=term>.

Member of classes:

DTD file:

Discussed in 13.2, “Tags for Terminological Data,” [in separate fascicle].

Discussed in 13.3.3, “Flat Term Entries Using Group and Depend Attributes,” [in separate fascicle].

Discussed in 13.4, “Overall Structure of Terminological Documents,” [in separate fascicle].

Class Name: **timed**

Description: elements which have a duration in time expressed either absolutely or by reference to an alignment map.

Attributes:

*start*: indicates the location within a temporal alignment at which this element begins.
Data type: IDREF
Value: contains the identifier of a previously defined **<loc>** element
Default value: #IMPLIED
Remarks: If no value is supplied, the element is assumed to follow the immediately preceding element at the same hierarchic level.
This attribute is optional.

*end*: indicates the location within a temporal alignment at which this element ends.
Data type: IDREF
Value: contains the identifier of a previously defined **<loc>** element
Default value: #IMPLIED

Remarks: If no value is supplied, the element is assumed to precede the immediately following element at the same hierarchic level.

This attribute is optional.

*dur* (i.e. duration): indicates the length of this element in time, using either specific units or the units specified on the associated temporal alignment.
Data type: CDATA
Value: contains a number optionally followed by a standard unit indicator
Default value: #IMPLIED

Remarks: If units are not defaulted, they should be represented using standard abbreviations (s for second, m for minute, etc.)
This attribute is optional.

Part: type=base name=spoken>.
Member of classes:
DTD file:
Discussed in 10.4.3.2, “Temporal Information,” [in separate fascicle].

Class Name: **xPointer** (i.e. extended-pointer elements)
Description: groups elements which use the TEI extended pointer mechanism to point at locations which have no SGML ID.
Attributes:

*doc* (i.e. document or file): specifies the document within which the desired location is to be found.
Data type: ENTITY
Value: The name of a system entity associated with the document within which the target of this extended pointer is to be found, by default the current document.
Default value: #IMPLIED
Example:

```
<xptr doc=Chap2 from='id (e23)'>
```

Remarks: The system entity named by this attribute must be declared in the DTD of the document containing the extended pointer.
This attribute is mandatory when applicable.

*from*: specifies the start of the destination of the pointer, as an expression in the TEI extended-pointer notation described in section 16.2, “External pointers and references,” [in separate fascicle].
Data type: %extPtr;
Value: The value specified must be a valid expression in the TEI extended pointer notation defined in section 16.2.1, “TEI extended pointer syntax specification,” [in separate fascicle].
Default value: #IMPLIED
Example:

Remarks: If no value is specified, the target is the whole of the document identified by the *doc* attribute.
This attribute is mandatory when applicable.

*to*: specifies the endpoint of the destination of the pointer, as an expression in the TEI extended pointer notation.
Data type: %extPtr;

Value: The value specified must be a valid expression in the TEI extended pointer notation defined in section 16.2.1, “TEI extended pointer syntax specification,” [in separate fascicle].

Default value: "DITTO"

Example:

```
<xptr doc=OrbisPictus from='id (animalia)' to='id (aquaticae)'>
```

Remarks: This attribute may only be supplied if the *from* attribute is also supplied, in which case the destination is defined to extend from the beginning of the location specified by the *from* attribute, up to the end of that specified by the *to* attribute. It is an error for the *to* attribute to specify a location whose end precedes the beginning of the location specified by *from*; it is not an error for the scopes to overlap.

If no value is specified, the target is the location specified by the *from* attribute.

This attribute is mandatory when applicable.

Remarks: This class belongs to the larger class *pointer*, which means its elements also inherit the attributes of that class.

Part: type=top name=xrefs>.

Member of classes: pointer

DTD file: dummy

Discussed in 16.2.1, “TEI extended pointer syntax specification,” [in separate fascicle].

# *Usability Report and Comment*

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

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

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

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

Your name:
Your postal address:

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

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

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

Thank you!

# *Temporary Index*