**Telecommunications Industry Forum**

**Sponsored by the Alliance for Telecommunications Industry Solutions**

# Telecommunications Interchange Markup, Part 1: Introduction to TIM

TIM Version 2

TCIF Guideline

TCIF-IPI-95-004-PART1

Issue 1A, July 1997 DRAFT

# Telecommunications Interchange Markup, Part 1: Introduction to TIM

Prepared for TCIF by the Information Products Interchange Committee. For more information about TCIF, go to http://www.atis.org/atis/tcif/. To order this document, please contact TCIF at (202) 434-8836, FAX (202) 393-5453.

If you have questions or comments about this document, please contact Donald F. Pratt, (732) 699-4012, dfp@ims.bellcore.com.

## Contents

## List of Exhibits

## List of Figures

# 1 Introduction

## 1.1 Purpose and scope of document

This document defines and describes the second major release of Telecommunications Interchange Markup (TIM 2), which can also be called Technical Interchange Markup. TIM is an implementation of Standard Generalized Markup Language (SGML) that has been designed specifically for interchange of technical documentation in the telecommunications industry. It is likely to be appropriate for technical documentation in other industries, and for some kinds of nontechnical documentation.

The Telecommunications Industry Forum (TCIF) has been developing voluntary industry guidelines for electronic document delivery since 1988, when the benefits of electronic information over paper — in availability, usability, and cost — were becoming apparent. All members of TCIF recognize the value of voluntary guidelines that avoid duplication of systems and effort in the operations of telecommunications service providers. This document provides such guidelines for the markup of electronic information products that may be provided in addition to or in place of paper documents.

If a document originator and a document recipient both use the same applications and file formats for documents, such as a particular version of MS Word, PageMaker, or HTML (HyperText Markup Language), they should feel free to exchange documents in those shared formats, but such a match will occur only occasionally — in fact, it is rare that everyone in the same company uses the same application. TIM and related guidelines provide an industry-wide shared format that can be used as is or converted for use in almost any application.

These guidelines are voluntary, subject to agreement of both the supplier and the recipient. They are an option, not a requirement. They have the advantages over other options of having been designed for interchange and having been tested by the participating members of the TCIF Information Products Interchange Committee. Moreover, many of the participants have made efforts to incorporate these guidelines into their documentation processes, for instance by developing authoring DTDs that translate easily into TIM.

The Information Products Interchange (IPI) Committee of TCIF chose SGML for exchange of technical documentation because:

- It is an international standard (ISO 8879).
- It is not dependent on any particular hardware, software, or presentation medium — it works equally well on various computers and can be used for paper, CD-ROM, or on-line documentation.
- It is supported by a large and growing number of products and applications that either use SGML files directly or can convert them to their own formats.
- It can be tailored to the needs of a particular industry.

- It structures the information content of documents in a way that maximizes the potential for reuse, repackaging, and repurposing.

IPI developed the TIM implementation of SGML specifically to meet the common needs of telecom equipment vendors who deliver documentation for their products and the carriers who receive that documentation. TIM is not intended to be used for authoring: Most originators will prefer more highly structured DTDs or semantic DTDs that constrain and therefore simplify creation of documents fitting their particular document designs. TIM identifies only the generalized structural components that occur in telecom-industry documents, yet it allows originators of the documents to pass on all useful metainformation about their specialized use of those components (structural and semantic labels, languages used, revision status, cross-references, keywords, etc.).

TIM is rigorously defined (from a computer's point of view) by an SGML DTD (Document Type Definition), provided in these guidelines. The DTD defines the markup that surrounds the content of the document: for instance, each paragraph is marked by **<P>** at the beginning and **</P>** at the end, and an italicized word is marked with **<Emph>** and **</Emph>**. Any part of a document that is not ASCII text is given an ASCII name and, if necessary, stored in a separate file: for instance, a bullet (•) is represented as "•" and a graphic file is given a name that points to where it is stored. With the TEDD packaging guidelines (TCIF-IPI-95-001) and TIM markup, there is a way to deliver every part of a technical document, even if it contains hypertext links, multiple languages, sound, video, or even a data type that hasn't been invented yet.

The benefits of TIM are not free. Using TIM requires tools and techniques that are not yet familiar to every telecom document provider and recipient. (If you aren't familiar with TIM but have used HTML, which is similar but simpler, you have an idea of what's involved.) It must be noted that it is harder for document providers to convert their existing word-processor or other non-SGML files into TIM than it is for recipients to convert TIM into their own formats, because SGML contains more information than most formats. Yet many document providers have already begun using TIM or other versions of SGML because they've seen that the benefits of well-structured, reusable information outweigh the costs.

This document is not a tutorial on SGML, and it does not explain every nuance of TIM. But it does have enough information in it that a person familiar with SGML applications could create valid TIM versions of new or existing documents. (For persons not familiar with SGML applications, tutorials have been developed.)

Related guidelines produced by the IPI Committee describe the recommended file formats for interchange of graphic and other non-text content of an electronic document. Document providers who package their electronic documents according to the full set of guidelines will be assured that most recipients will be able to use them as intended without special instructions, consultations, or technical support. Recipients prepared to receive these packages will be able to follow the same steps to distribute, store, and use documents from all participating providers. This means in particular that it will

be possible to use the same hardware, software, and procedures to view documents from all providers.

Neither these guidelines nor any others under development by the IPI Committee specify a delivery medium for documents. Means of delivery (tape, diskette, CD-ROM, modem, Internet, etc.) will always need to be agreed upon by originator and recipient.

Issue 1A of these guidelines is a draft for review and comment during the testing and balloting for TIM 2. Having been adopted, these guidelines are being published as Issue 1.

## 1.2 Organization of this document

Section 1 provides an introduction to this document.

Section 2 is an introduction to SGML and TIM.

Section 3 describes the components of a TIM document ("elements") and how they are marked up. It is the starting point for creating a TIM document.

Section 4 describes the additional information contained in TIM markup ("attributes," "entities," and "notations").

Section 5 is about what you need to do as a producer or user of TIM documents. It includes information on how to get updated files and documentation, free or nearly free software that works with TIM, answers to questions about using TIM, and a response to your views on how to improve it.

Section 6 is a verbatim listing of the TIM DTD and related files. It's better to get the files themselves from one of the TCIF repositories:

- **`http://www.atis.org/atis/tcif/ipi`**
- **`ftp://ftp.bellcore.com/pub/world/TCIF`** (command-line ftp only)
- **`ftp://ftp.isogen.com/pub/tcif`** (accessible with Web browsers)

TCIF-IPI-95-004-PART2 is an alphabetical guide to the TIM elements and attributes, with advice on how to use each one. It is the experienced user's reference.

## 1.3 Referenced standards

These guidelines are based on the following standards:

ISO 8879, *Information processing — Text and office systems — Standard Generalized Markup Language (SGML)*. TIM is an SGML Document Type Definition — that is, a particular implementation of SGML.

ISO/IEC 646, *Information technology — ISO 7-bit coded character set for information interchange*. All SGML files use the G0 character set of ISO 646, equivalent to ASCII (binary data must be stored in separate files).

ISO 639, *Codes for the representation of names of languages*. Languages used in TIM files are identified according to this standard.

## 1.4 Related documents

TCIF-IPI-95-004-PART2 is the alphabetical reference to TIM. TCIF-IPI-97-004 is a short summary of the differences between TIM 2 and TIM 1.

TCIF-IPI-95-001 describes Telecommunications Electronic Document Delivery (TEDD) package, which specifies the directory structure, file-naming options, and identifying information for the complete document package that includes a TIM file. Future IPI documents are expected to deal with document identifiers and with admonishments (warning messages).

TCIF-IPI-96-004 specifies the acceptable formats for raster graphics ("bitmaps") delivered as part an electronic-document package. Guidelines for vector graphics and multimedia objects are planned.

## 1.5 Use of product names

It is contrary to the intent of this document to provide any support whatsoever for particular products of, or practices favored by, some manufacturers and suppliers over others, except to support in a general way those that are consistent with the standards referred to above. Names of commercial products are used in this document for illustrative purposes only. No endorsement or evaluative judgment of any product, service, or provider is intended, and none should be inferred.

## 1.6 Acronyms

The following acronyms are used within this document:

**ASCII:** American Standard Code for Information Interchange, which defines the 7-bit encodings of the characters seen on a standard computer keyboard, plus control characters (128 characters in all). In this document ASCII is also referred to as ISO 646. TIM files must contain only ASCII characters.

**DTD:** Document Type Definition, a specification for the SGML markup of a particular type or class of document. These guidelines contain the TIM DTD, which specifies the markup recommended for technical documents interchanged (in TEDD packages) within the telecommunications industry.

**HTML:** Hypertext Markup Language, an informal standard for markup of documents accessible on the World Wide Web. Some versions of HTML have been defined in SGML DTDs. HTML is oriented toward general-purpose browsers, so it does not provide the structural and semantic detail found in specialized DTDs like TIM.

**ISO:** International Organization for Standardization.

**SGML:** Standard Generalized Markup Language, an international standard (ISO 8879) for markup of text components of documents. Because it is rigorously

defined, SGML allows automated translation into other markup languages, rigorous or not; conversely, translation into SGML from other markups usually requires some guesswork and cleanup.

**TEDD:** Telecommunications Electronic Document Delivery, describing the packaging specified in a separate TCIF guideline (TCIF-IPI-95-001) for all the components of a document, including the TIM file.

**TIM:** Telecommunications (or Technical) Interchange Markup, the SGML DTD developed by the TCIF IPI Committee and recommended by them for markup of the text in documents interchanged in TEDD packages. It is defined and described in this document, with further details in TCIF-IPI-95-004-PART2.

**TIMM:** Telecommunications Interchange Markup (Minimal), a streamlined version of the TIM 1 DTD. Since the TIM 2 DTD is already much shorter than TIM 1, there is no longer a TIMM version.

# 2 SGML and TIM

## 2.1 Structured documentation

When documents are created with a word-processing or desktop-publishing program such as Word or FrameMaker, special codes are inserted into the document that provide information about how the text is to be formatted (font, type size, line justification, etc.). This markup describes the way the document should look when printed or displayed on a screen, but does not provide any explicit information about the document itself (What type of document is it? What kinds of information does it contain?).

Appearance-oriented markup is fine when the only product is paper. Readers can usually infer most of what they need to know about a document by the way it looks. We know, for example, that a line of bold text that begins with the word "Chapter" is a chapter title, and we can infer that a new chapter has begun. We also know that a distinct block of text is generally a paragraph; and that italicized text represents some form of emphasis. We recognize these structures easily without being told what they are because we've learned the conventions that relate a document's appearance to its structure.

Appearance-oriented markup does not work well, however, in electronic publishing, because documents generally have to be reformatted to be read on-screen, and computers are exceptionally poor at discerning the structure they must preserve from the appearance-oriented markup they must discard. A computer must be told explicitly what each structural element is.

Suppose the computer did simply reproduce a document's appearance without understanding its structure, leaving the interpretation to the reader as is done in paper publishing. Besides reducing readability, that removes many of the offsetting advantages electronic publishing has over traditional paper publishing. Why, for instance, should the reader settle for "see page 5-17 for more"? A computer that could recognize a cross-reference in the text could turn it into a hyperlink: "click here for more."

Electronic documents in any form have advantages over paper in speed and ease of delivery. Going beyond those, one of the best reasons for electronic publishing is that it lets the producer reuse and repackage information with little or no added cost. A single electronic document can be used to print a paper copy, create a colorful hypertext CD-ROM product, and load information into a database for on-line search and retrieval using low-cost terminals. And it can be rendered in large type or through text-to-speech devices for the sight-impaired. This is possible because the computer can apply different formatting rules to the same document depending on the output medium, as long as it understands what kinds of information it is dealing with. For these things to happen, the structure of the document must be explicitly defined and understood by the computer.

A further advantage to structural markup is its vendor- and platform-independence. It is easy to write formatting rules like "make all chapter titles

18-point Helvetica bold, and have them start a new page." And it is easy to translate such rules into the appearance-oriented markup of any word processor, desktop-publishing system, or web browser.

## 2.2 SGML

SGML (Standard Generalized Markup Language) is an ISO standard for creating structured documents in a vendor- and platform-neutral manner. SGML is used to define the structural elements that make up a document (such as chapters and paragraphs); the relationships that exist between those structural elements (e.g., a chapter contains one or more paragraphs); and the attributes that each structural element can possess (for instance, chapters and paragraphs can be numbered or unnumbered). All value-added information contained within an SGML file relates to the structure of the document. An SGML file does not specify anything about the appearance (or output medium) of the document. Therefore, a single SGML file can be rendered in a variety of ways according to formatting rules appropriate for a given output system and application.

The structure of any particular SGML document is defined explicitly in a Document Type Definition (DTD). A DTD is a series of declarations that define the structural elements of the document, their content models (what elements they can contain, in what order), and the attributes that can be assigned to the element. In practice, the DTD is a set of rules for building a structured document.

## 2.3 TIM SGML

TIM (Telecommunications Interchange Markup) is an SGML DTD, a blueprint for documents interchanged in the telecommunications industry. TIM has been designed by the TCIF Information Products Interchange Committee to capture the structure found in the kinds of technical documents that are interchanged among telecom companies. Whether a company uses TIM files directly or converts them to something else, TIM will guarantee that technical documents will always be in a consistent format, with an explicit structure that can be readily understood and processed by electronic publishing systems.

Because TIM documents are plain ASCII files with a standard structure and coding scheme, they are much easier to convert into word-processor and other formats than the word-processor formats are themselves, since those have proprietary features and codes that are not only system-specific but often inconsistent from release to release.

Some of the benefits of using TIM as an industry-wide interchange format are:

1. **Lower publishing costs**. A vendor who is distributing electronic documents to several end users will have to distribute only one file format, thereby eliminating the need for extra conversion and cleanup steps.
2. **Multiple output formats**. A single TIM file can be output in a variety of formats (paper, hypertext CD-ROM, etc.) without additional coding and with

minimal conversion costs, making it easy for vendors to produce low-cost custom products.

3. **Reuse of information**. Because the value-added information within a TIM document relates to its content and not its appearance, TIM documents can be stored as a flexible database of information that can be repackaged and reused in a variety of ways. For example: TIM documents (or portions of them) can be embedded into other corporate documentation; portions of several TIM documents can be combined to create new documents; and TIM documents can be linked so that the information in several related documents is updated or revised simultaneously.

## 2.4 Inside SGML

TIM is an SGML application. To understand how TIM works, it is necessary to understand the basics of SGML. SGML is an international standard that provides a methodology for inserting value-added information into a document regarding its structure and content through the use of plain ASCII tags. Any SGML application like TIM specifies what the tags are and how they're related to each other. It's assumed that the tags make sense for only a certain kind of document, hence the specification is called a Document Type Definition. The DTD is the core of an SGML application. The application also needs an SGML declaration, which defines some basic formatting rules, and of course a document instance, a file with TIM markup. Besides those three basic ingredients, end users must also have a style sheet or processing instructions that tell their particular hardware and software how to output the SGML document in usable form, but since that is platform-dependent, it is not discussed here. The document instance is all that most people will see.

### 2.4.1 The SGML document instance

An SGML document instance is a specific instance of a document class that has been defined by a DTD. If a DTD defined a document class called "memo" for a given company, then an SGML-tagged memorandum generated within that company would be a "memo" document instance.

Take, for example, the following departmental memo:

**Exhibit 2-1** Sample Memo

---

MEMORANDUM

**TO:**John

**FROM:**George

**DATE:**1/1/95

**RE:**Today's meeting

Today's meeting has been canceled due to extenuating circumstances beyond my control. It has been tentatively rescheduled for tomorrow at the same time. Please let me know if this is okay with you.

I apologize for the inconvenience.

---

In SGML, the same memo might look like this:

**Exhibit 2-2** Sample SGML Instance of Memo

---

```
<MEMO type="departmental">
<HEADER>
<TO>John</TO>
<FROM>George</FROM>
<DATE>1/1/95</DATE>
<RE>Today's meeting</RE>
</HEADER>
<BODY>
<PARAGRAPH>Today's meeting has been canceled due to
extenuating circumstances beyond my control. It has been
tentatively rescheduled for tomorrow at the same time. Please
let me know if this is okay with you.</PARAGRAPH>
<PARAGRAPH>I apologize for the inconvenience.</PARAGRAPH>
</BODY>
</MEMO>
```

---

SGML captures the structure of a document through the use of ASCII tags (surrounded by angle brackets) inserted into the text of the document. Each structural element is enclosed in corresponding start and end tags. Start tags begin with "<" and end tags begin with "</". Structure is shown by nesting elements inside each other.

Note that the SGML document instance does not contain any information regarding how the document is to be displayed. The bolded words **MEMORANDUM**, **TO:**, **FROM:**, **DATE:**, and **RE:** do not even appear in the text of the document instance.

Because the words do help the reader understand the document, however, the processing system could insert these words when printing the document. It could do so by applying a rule such as: **print "TO:" before each <TO> element in a 'memo' document**. In this manner, an SGML file can be printed just like a word-processing file.

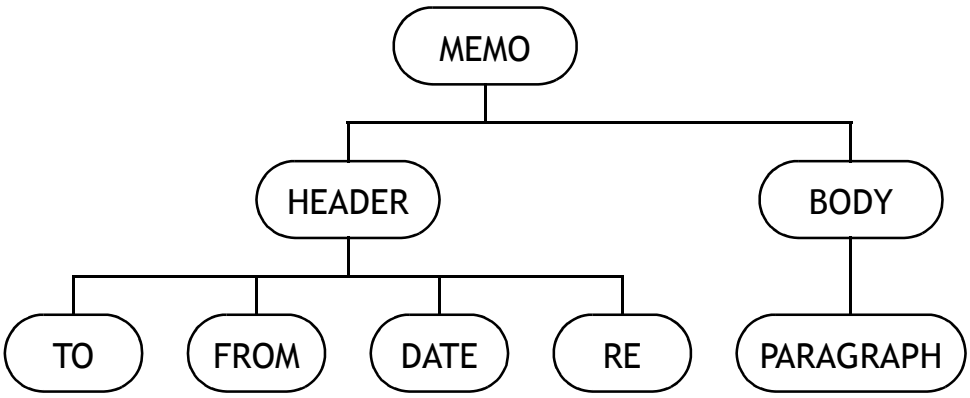
The processing system could generate a different appearance by applying different formatting rules to the same element. For example, the processor's rule for displaying the same document on-line could be: *display graphic icon "to.gif" before each <TO> element in a 'memo' document*. This flexibility is one of the biggest advantages of SGML; it allows information in documents to be repackaged and reused without the cost of manual reformatting.

Once a document instance has been fully tagged, it is usually processed by a piece of software called an SGML parser. The parser checks the tags in the document instance against the structural rules in the DTD. Any markup that does not conform to the rules of the DTD cannot be processed automatically and must be resolved by the originator.

## 2.4.2 The Document Type Definition (DTD)

The sample memo has this tree structure:

**Figure 2-1** Document Structure of a Memo




In SGML the structure is prescribed by the Document Type Definition. The Memo DTD may, for instance, require that all four parts of the header be present in fixed order, and that the body contain at least one paragraph.

Exhibit 2-3 is a hypothetical Memo DTD:

**Exhibit 2-3** Hypothetical Memo DTD

```
<!ELEMENT memo      - - (header, body) >
<!ATTLIST memo  type    (departmental|interoffice|
                         external) "departmental" >

<!ELEMENT header    - - (to, from, date, re) >
<!ELEMENT to        - - (#PCDATA) >[1]
<!ELEMENT from      - - (#PCDATA) >
```

```
<!ELEMENT date       - - (#PCDATA) >
<!ELEMENT re         - - (#PCDATA) >


<!ELEMENT body       - - (paragraph+) >
<!ELEMENT paragraph - - (#PCDATA) >
```

The DTD is an ASCII file that contains "declarations" defining each structural element's generic identifier, content model, and attributes, as described below:

- Generic identifier — This is the tag name used to identify the element within an SGML document. For example, a "paragraph" element may have the generic identifier "paragraph." Within an SGML document, paragraphs would then be marked with the tag **<paragraph>**, as in Exhibit 2-2. (TIM paragraphs are marked **<P>**. Case is ignored for all markup except entity names.)
- Content model — This is the definition of what type of data can be contained within the element. For example, a "paragraph" element could be allowed to contain plain text (which the DTD refers to as "#PCDATA"), emphasis, lists, figures, and tables. The content model indicates what elements can occur inside a given element, the order (if any) in which they can occur, the number of times they can occur, and whether they are required or optional. For instance, the commas in the declaration for "header" mean that the components must occur in order, and the "+" in the declaration for "body" means that it must contain at least one paragraph. In this manner, the content models provide all the structural rules for a document.
- Attributes — An attribute assigns a particular value to an element. A company may have several types of memos (e.g., departmental, interoffice, external, etc.). Therefore, a "type" attribute with a value of either *departmental*, *interoffice*, or *external* would be assigned to the "memo" element of this document to distinguish it from other types of memos. For the sample memo, the value "departmental" was chosen (Exhibit 2-2).

The declarations that begin with "<!ELEMENT..." are element declarations. An element declaration contains the element's generic identifier (tag name), followed by the content model (in parentheses). The declaration that begins with "<!ATTLIST..." is an attribute list declaration. It defines the attributes that can be assigned to a given element, the possible values for each attribute, and the default value (if any) of each.

In addition to element-specific declarations, the DTD may contain entity declarations. In SGML, an entity is an information object that is not contained directly within the SGML document instance because it contains non-ASCII code, requires separate processing, and/or conforms to a different DTD. Such entities include non-ASCII special characters (such as bullets, diacritical marks, and

1. "#PCDATA" means "parsed character data." It does not contain any further subelements.

Greek symbols); graphic, audio, or video files; compilable source code; etc. The DTD (including part of it that is specific to each document) indicates the entities that are allowed to be referenced within the document. An entity declaration for an external graphic looks like this:

```
<!ENTITY figure1 SYSTEM "../tiff/fig001.tif" NDATA TIFF>
```

This declaration has an entity name (to be used later in the document, where the graphic should appear), a system-specific identifier (there are also "PUBLIC" identifiers), and a specification of the file format of the graphic (because you may need to use different applications to view different kinds of graphics).

### 2.4.3 The SGML declaration

The SGML declaration is an ASCII file that provides the processing system with the basic syntactic information it needs to interpret the SGML application. This file is used only by the SGML processing system or parser. The end user of an SGML document does *not* need to understand the SGML declaration. It includes such things as the character set to be used (TIM uses ASCII, but other character sets, such as UNICODE, are possible); which characters are reserved for SGML markup (e.g., "<" is used as a tag delimiter, so the "<" entity reference must be substituted for a "<" in the document text); and whether some markup can be omitted where there would be no ambiguity (in TIM it cannot be).

## 2.5 HTML, SGML, and TIM

While most people have never heard of SGML, many are now becoming familiar with its concepts through the use of Hypertext Markup Language (HTML). When learning about SGML for the first time, people often ask, "Why not just use HTML instead, since it is so widely accepted?" To understand the answer to this question, it is necessary to first understand what HTML is, what it is intended for, and how it falls short of SGML for many electronic-publishing applications.

HTML 2.0 and 3.0 are valid implementations of SGML, but they follow only some of the principles of SGML and provide only some of the benefits. Like TIM, HTML uses generic ASCII tags to mark up documents according to rules defined in a DTD. Unlike TIM, however, tags in HTML do not represent the structure of a document (except in a trivial way, because virtually every structural element is defined so that it can contain any other structural element). In practice, HTML tags merely define how the document is to be displayed on a World Wide Web browser (such as MOSAIC or Netscape). Because HTML files contain mainly display-oriented markup, they lack many of the advantages of structured SGML files. Elements within an HTML document have no defined relationship to each other, just as in a word-processor file; therefore, a publishing system cannot apply generic publishing rules to an HTML file because it cannot readily infer the document's structure.

Even if HTML could be used to create structured documents, it would still be inadequate for most publishing applications. The HTML DTD defines a very simple (and relatively short) type of document. Because the HTML DTD is "hard-wired"

into WWW browsers (unlike SGML software, which can read in any DTD), it cannot be modified by users to accommodate more complex documents. Many structural elements that occur in telecommunications documents (such as procedures, requirements, and danger/caution/warning messages) cannot be adequately represented by the tags available in HTML. Of course, TIM files can be converted to HTML for end users who have only web browsers to work with, with considerable loss of information in the markup but no necessary change in appearance.

TIM is an SGML application that is fully compliant with international standards. TIM documents can be read and processed by any SGML-compliant software regardless of the software manufacturer or computer platform. Software that works with SGML files directly (or nearly so) includes ArborText PublisherSGML, EBT DynaText, FrameMaker+SGML, Interleaf World View SGML, Microsoft SGML Author, MicroStar Near & Far Author, SoftQuad Panorama and Author/Editor, and WordPerfect SGML. TIM files can be converted to HTML, RTF (Microsoft Word), and potentially any other format needed by publishing applications that are not SGML-compliant.

Using TIM does not mean no more conversions, or even no more loss of information in conversions, but with good programming it does mean no more cleanup after conversion and no more hanging on to obsolete hardware and software because conversion is too costly.

# 3 TIM Elements and Structure

This section introduces the specifics of TIM document structure. In an SGML file, all text and other content is enclosed in structural units called elements. These in turn are enclosed in higher-level elements until, at the top of the structure, one element encloses the entire document. The most noticeable way that one SGML DTD, like TIM, differs from another is in what kinds of elements are defined. What matters is not the name of each element but its content model. Elements are defined from the top down: each element's content model says what kinds of data or subelements it can contain, whether they're required or optional, and in what order they can appear.

## 3.1 A Simple Example

Exhibit 3-1 shows a minimal TIM file (not quite the absolute minimum, because it contains one paragraph of real content, which is not absolutely necessary):

**Exhibit 3-1** Simple TIM document instance

```
<!DOCTYPE TDoc PUBLIC "-//USA/TCIF//DTD TIM-2//EN">
<TDoc>
  <DocID>
    <EdocID>TCIF-IPI-95-004:issue0:instance1</EdocID>
    <Class>TEDD-Package-3.0</Class>
    <Company>TCIF</Company>
    <DocNo>
      <DN.base>TCIF-IPI-95-004</DN.base>
    </DocNo>
    <DocDate>1995-12</DocDate>
    <TitleGroup>
      <Title>Telecommunications Interchange Markup (TIM)</Title>
    </TitleGroup>
    <MTextType>TIMM-2.0.0</MTextType>
  </DocID>
  <Body>
    <P>Hello world!</P>
  </Body>
</TDoc>
```

Most of the lines in this minimal file are required elements of the document-identification section, which is exactly the same as the *docid* file in a TEDD package (see TCIF-IPI-95-001), except that a TIM version number is also required (in the **<MTextType>** element). The **<Body>** subdivision is also required. Beyond the minimum, a TIM file may have more elaborate identifying information, including issue, edition, revision, supplement, and other such numbers, a copyright notice, authors, publishers, an ISBN number, an abstract, keywords, and more. Then it may have frontmatter, appendixes, and backmatter in addition to the body. Those major divisions may be divided into any number of levels of sections, or not, and may also contain nonhierarchical structural units called admonishments ("Caution," "Danger," and "Warning" messages), annotes (notes, footnotes, comments, etc.), tables, graphics, and other things. Five kinds of lists are defined, not counting variations in numbering/lettering/mark style.

Paragraphs and other text elements can contain subscripts, superscripts, emphasized text, symbols, and text labeled as special for other reasons (quotations, filenames, document titles, etc.). Tables, graphics, multimedia objects, equations, and code listings all have their appropriate enclosing elements. Cross-references can be expressed as HyTime or URL (Uniform Resource Locator) hyperlinks.

Other features allow portions of the document to be in separate pop-up windows or to be provided in more than one representation (different languages, different graphic file formats, etc.). Tables of contents and indexes are not normally provided with electronic documents because most applications have full-text search capabilities, but they can be included in TIM, or specifications for them can be given.

Because TIM is meant for interchange of all kinds of technical documents to all kinds of users, it deliberately avoids defining elements that have narrow uses. A DTD specifically for a parts catalog, for instance, might have elements named **`PartNumber`**, **`Description`**, and **`Price`**. But a DTD with all the specific tags that could be used in all technical documents would be much too complicated for general use. Fortunately, it is easy both in theory and in practice to convert files from one SGML DTD to another, so it can be cost-effective to use TIM for interchange while adopting more specific DTDs for specific uses.

## 3.2 Elements and structure

The names of the elements in an SGML file provide some information about the content of the document, but only because the elements are given names that make sense to people. An SGML application won't care if the headings are called **`<P>`**'s and the paragraphs are called **`<Head>`**'s. For the application, the element definitions (known as declarations) that go with the names are all that matter, and their only role is to define the structural rules for the elements: for each element, starting the with the one that encloses the whole document, the definition sets the rules for what other elements can occur inside it, and in what order. These rules are called the element's "content model."

An SGML file usually does provide more information than just the structure of the document. Any kind of information that is useful can be included in attributes, which are predefined variables with either a limited or an unlimited set of possible values, according to the attribute declaration for each element. Attributes defined for TIM allow elements to be numbered, linked, categorized in at least four ways, assigned keywords and revision status, emphasized, indexed, and re-encoded with tags they had in another markup. Attributes are specified by putting them inside the element's start-tag:

```
<P emph="bold">Hello world!</P>
```

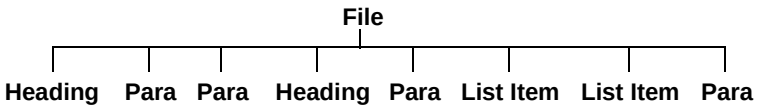
Almost everything you will see in a TIM file is either content or part of an element start- or end-tag, and inside the tags everything that is not the name of the element is an attribute. TIM attributes are discussed in Section 4. For now it is enough to know that the first word in an element's start-tag is the element's

name (known formally as its generic identifier, or GI), and that this is necessary and sufficient to determine the element's place in the document structure.

## 3.3 Basics of structure

One of the main distinctions between a TIM document and a typical word-processor file is that a TIM document contains a rigorously defined structure, while a word-processor document is unstructured or "flat." In an unstructured document, text elements (such as chapters, titles, paragraphs, etc.) have no relationship to each other except that they occur in a given order, as shown in Figure 3-1.

**Figure 3-1** Unstructured document diagram

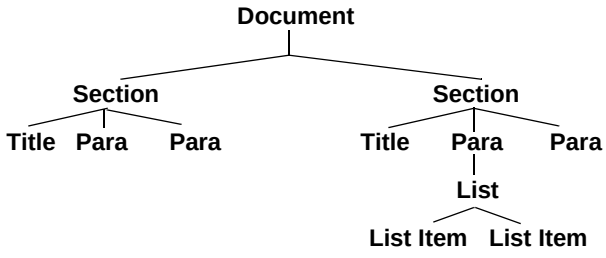



Or, as you might see it in your word processor, if you set y document window to show "styles":

**Heading**
**Normal**
**Normal**
**Heading**
**Normal**
**ListItem**
**ListItem**
**Normal**

In structured documents, the hierarchical relationship between elements is clearly defined. This hierarchical structure can be illustrated in a tree diagram, as in Figure Figure 3-2.

**Figure 3-2** Structured Document Diagram



The structure of TIM documents is formally defined by the TIM DTD (document type definition), in Section 6. The TIM DTD file is intended for use by SGML application software, and people who are not very familiar with SGML may find it difficult to read. This section describes the document structure defined in the TIM DTD in mostly nontechnical terms. For instance, the terms used in this section to describe the hierarchical relationship between elements in a TIM document are mostly words that make it sound like we're talking about a family tree, although in an SGML tree structure each element has only one "parent":

**parent:** the element directly containing a given element. In Figure 3-2, "Para" is the parent of "List."

**child:** an element contained directly within a given element. In Figure 3-2, "Section" is a child of "Document."

**sibling:** an element that is on the same hierarchical level as a given element, and has the same parent. In Figure 3-2, "Title" and "Para" are siblings, but the two "Titles" are not.

**ancestor:** an element that is one or more hierarchical levels above a given element. In Figure 3-2, "Document," "Section," "Para," and "List" are all ancestors of "List Item."

**subelement:** an element that is one or more hierarchical levels below a given element. In Figure 3-2, "Para," "List," and "List Item" are all subelements of "Section." For some reason, we don't say "descendent."

The structure of a TIM document can be broken down into three main levels:

1. **Superstructure level.** This level represents the organizational structure of the document. It includes the document's identifying information and the hierarchy of organizational divisions: frontmatter, body, appendix, and backmatter, each of which may be divided into any number of levels of sections. If the superstructure were removed, the content would be about the same, although it would be harder to navigate.

2. **Base level.** This is the familiar level at which you find the content of the document. It includes paragraphs and other things that can appear where paragraphs can appear, but which may be more elaborate structural units, called "Requirements," "Procedures," "Warnings," etc. These structures are subdivided into paragraphs, so in the end almost all ordinary text in a TIM file, other than what's in headings of various kinds, is enclosed in **<P>** paragraphs. Much of the content may not be subdivided any further than this, but this much structure is always present — if only because we don't have authoring tools that *don't* create paragraphs.

   It is possible in some DTDs for paragraphs to contain subparagraphs. TIM does not have subparagraphs (although there is always a **<P>** element within a list item, which may be within a **<P>** element). This is a deliberate simplification that makes it easier to convert between TIM structure and the completely flat structure of word-processing files, where most TIM files are created and updated.

3. **Substructure level.** This level represents structures occurring sporadically within the paragraphs and paragraph-like structures. It includes most lists, tables, and figures (although these are allowed to occur at the base level as well as within paragraphs) and phrase structures like filenames, cited document titles, and subscripts. This structure can be as rich as you wish it to be: there is no limit on the kinds of information that can be specially tagged (part numbers, descriptions, and prices, for instance) or on the number of levels of nested subelements.

   (How can part numbers be tagged if there's no tag for them, as was mentioned before? There's no tag *specifically* for part numbers, but with a combination of general-purpose tags and user-defined attribute values, any specific kind of information can be identified.)

There is usually also some **metastructure**, consisting of cross-references, hyperlinks, indexes, tables of contents, and other such things that provide information about hierarchical and nonhierarchical relationships within the document and between documents.

We'll discuss these structural levels in the order you're likely to notice them looking through a TIM file.

## 3.4 Superstructure level

A TIM document contains several elements that define how the document is organized and provide important identifying information regarding the document. (Identifying information is considered to be part of the superstructure because its main use is to place the document into an even larger structural context.)

The top-level element in a TIM document, containing the whole document, is the **`TDoc`** element. **`TDoc`** comprises six main subelements:

1. Document identification, **`DocID`**, which is required
2. **`Resources`**, which is optional
3. **`FrontMatter`**, optional
4. **`Body`**, required
5. Appendixes, **`AppMatter`**, optional
6. **`BackMatter`**, optional

The last four of these main subdivisions can be further subdivided into **`Section`**s (3.4.7). The most important reason for this first set of subdivisions is that the main sections within frontmatter and backmatter usually have standard names ("Preface," "Table of Contents," "Glossary," "Index," etc.) but are not numbered, while body sections are likely to be numbered (e.g., "Chapter 1") and appendix sections lettered (e.g., "Appendix A").

### 3.4.1 <DocID> (Document Identifier)

The **DocID** is a required element that must appear at the beginning of the **TDoc** instance. The **DocID** contains all the information that uniquely identifies the **TDoc** instance, plus any other information useful in describing and categorizing it. **DocID** typically contains all the information that would appear on a title page and copyright page, but keep in mind that there really aren't any pages in an SGML document, only in some particular renderings of it (mostly the paper ones).

At a minimum, the **DocID** should consist of:

1. A unique electronic document ID (**EdocID**). This identifier must be different for each instance of the document, even, say, a minor correction that does not have a different issue or revision number.
2. The document instance class (**Class**). This is the class identifier (version number) of the TEDD package containing the **TDoc** instance. The **Class** element always (so far) says "TEDD-Package-3.0".
3. Company name (**Company**). This is the common name of the document's owner.
4. Document number (**DocNo**). This is a formally structured document identifier. The **DocNo** must contain a "base document number," called **DN.base**. Following the **DN.base**, the **DocNo** may contain any of several other "DN..." number components.

   An unstructured document number, called **DocNo.u** may be used instead of the **DocNo**, but **DocNo** is preferred.
5. Document publication date, **DocDate**.
6. Complex title (**TitleGroup**). Within the **DocID**, the **Title** within **TitleGroup** is the title of the document.
7. TIM version number (**MTextType**). It should always have the form "TIM-2.0.0".

In addition to these required elements, there are several optional elements that may occur within a **DocID**:

- **VersionControl**, containing any number of **Version** subelements identifying different versions of the document if they are combined in the same instance. Each **Version** subelement may contain **Au** (author), **VersDate** (version date), and **VersDesc** (version description) subelements.
- **AltNo** (any number), an alternative identifier, such as a part number
- **Publisher**, if different from originating Company
- **ISN** (any number), ISBN/ISSN number
- **Country** (any number), the country the document is written for
- **Copyrt**, copyright information

- **Lang** (any number), language code
- **Superdoc** (any number), superdocument
- **SubDoc** (any number), subdocument
- **CDocType**, file type of the "canonical document" included in the same TEDD package (typically PDF or PostScript)
- **DraftStatus**, "Draft", "Final", etc.
- **Status** (any number), any further information on the status of the document
- **PropStat**, proprietary status, and **PropMsg**, an explanatory message
- **EDocRepl** (any number), electronic document instance replaced
- **DocRepl** (any number), document replaced
- **ProdDsc** (any number), product description
- **DocDsc**, document description
- **RelDoc** (any number), related document
- **OrderInfo**, how to order the document
- **Contact** (any number), whom to contact for more information
- **Au** (any number), author
- **Abs**, abstract
- **Keywords**

### 3.4.2 <Resources>

The **Resources** section contains elements that have no fixed place in the document. They all have to do with hyperlinks: **Links**, **DataLocs**, **TreeLocs**, and **NameLocs**. See ****.

### 3.4.3 <FrontMatter>

The **FrontMatter** is an optional subdivision that may occur before the body of the document. The frontmatter typically contains any legally required disclaimers and copyright acknowledgments, and some guide to the contents. It may also contain background information on the document's history and intended audience, and other sorts of prefatory information.

The **FrontMatter** may be made up of one or more paragraphs (3.5.1) or other base-level elements. If it is long or complex, it may be subdivided into **Sections** (3.4.7).

### 3.4.4 <Body>

The **Body** is the only required subdivision of the document content. As the name implies, the body subdivision contains the body, the substantive portion, of the

document. The **Body** is typically made up of one or more **Sections** (3.4.7), but that is not necessary.

### 3.4.5 <AppMatter> (appendixes)

The **AppMatter** is an optional subdivision that may occur after the body. This subdivision contains appendixes to the document. Appendixes are special sections that explain or supplement the information contained within the body of the document. The **AppMatter** is typically broken down into sections, each child section representing a separate appendix to the document. If, however, there is only one appendix in the document, then the entire **AppMatter** can be considered a single appendix.

### 3.4.6 <BackMatter>

The **BackMatter** is an optional subdivision that occurs at the end of the document. This subdivision typically contains unnumbered sections such as glossaries, indexes, references, and bibliographies.

### 3.4.7 <Section>

The **Section** element can be used to subdivide the **FrontMatter**, **Body**, **Appmatter**, and **BackMatter** elements into any number of hierarchical levels. A section may be either numbered or unnumbered, and either titled or untitled. If titled, the section begins with a **Title** or **TitleGroup** (complex title, 3.4.8). A section can contain any number of base-level elements: paragraphs (3.5.1), admonishments (3.5.2), annotations (3.5.3), lists (3.5.4), distributed-list items (3.5.5), and frames (3.5.7). It may also contain subsections that can contain all these same things, and specially defined subsections for contents and index lists (3.7.2.6).

In some DTDs, the hierarchical level of each section is part of the element name (**Sect1**, **Sect2**, etc., instead of just **Section**). This was deliberately avoided in TIM because it is expected that both originators and recipients will restructure documents frequently. Section 3.5.1 of one document may become Section 2.4 of another. With TIM, the section can simply be moved; it does not need to be retagged. The trade-off is that when the sections are numbered, or point sizes are chosen for their titles, the nesting levels must be counted. Most SGML-aware applications can do that.

### 3.4.8 <TitleGroup> (complex title)

The main subdivisions of **TDoc** and their sections normally have titles, and they can also have one or more supertitles or subtitles, as well as a short version of the title. If there is more than a simple title, the various title elements are enclosed in a **TitleGroup**. Titles are optional except in **DocID** (the document must have a title), and when **TitleGroup** is used, the only required content is the **Title**. All other **TitleGroup** elements are optional.

1. **SuperTitle**. Used to specify a "family" of documents to which the document instance belongs, or to repeat a higher-level section title (for subsections). There may be more than one.
2. **Title**. Contains the title of the element. The **Title** does *not* contain the numbering for numbered elements (e.g., sections or list items). Such numbering is an attribute of the element to which the **Title** belongs.
3. **SubTitle**. A subtitle of the main title. There can be more than one.
4. **ShortTitle**. A short form of the title, for such things as running headers of pages, tables, etc., or page-number prefixes.

All of these elements have the same fairly basic content model, which allows text and a few substructure elements that might occur in a title, like subscripts and equations. Title elements do not have IDs, which means they cannot be the targets of cross-references. (This seems wrong to people who've used cross-references in word-processing documents, until they realize that their references are really to sections, tables, etc., not just their titles.)

Many of the base-level elements in TIM can also have titles: frames (which hold listings, graphics, etc.), lists, and other such structural units. There can also be titles over the columns of a list, but these are enclosed in a **ListHead** element rather than a **TitleGroup**.

## 3.5 Base-level structure

Almost all of the textual content of a TIM document is in **P** paragraph elements, but some of these are nested in more specialized structural units like lists, annotations, and admonishments.

### 3.5.1 <P> (paragraph)

**P** is the basic structural unit of a TIM document. It is used for paragraphs and paragraph-like structures. A paragraph can be numbered but not titled; if a title is required, the paragraph must be enclosed in something else, like a **Section** or an **S** structure. A paragraph contains running text (i.e., words and sentences). There are several text-level elements (such as those for superscripts and subscripts, emphasis, and symbols, 3.6) that can be used within paragraphs to convey special meaning or presentation.

In addition to running text, a paragraph can also contain most of the same structural groupings that can contain paragraphs, listed below. A paragraph within another structure has the same content model as one that is not, permitting information to be arranged in outline-style lists nested to any desired depth.

### 3.5.2 <Danger>, <Caution>, <Warning>, and <Admon> (admonishments)

Admonishments are safety messages, warnings of risks to be avoided in performing a described procedure. Admonishments are important in telecom

documents. They generally apply to one or a few steps in a procedure, and ideally they should be visible whenever any of the steps they apply to is visible, on a page or on a screen. TIM defines elements for the three types of admonishments that are commonly used in the telecom industry, plus a general-purpose element than can be made specific through use of attribute values.

Traditionally in telecom documents, a **`Danger`** message warns of the risk of serious or fatal injury or illness, A **`Caution`** warns of the risk of possible service interruption or of less-serious injury, and a **`Warning`** warns of the risk of equipment damage, software corruption, or data loss. (These meanings are not entitrely consistent with ANSI standards and OSHA regulations, and the TCIF IPI Committee will be seeking to define consistent uses and meanings for admonishments in the near future.) The admonshment elements have the same content models as an **`s`** structure (the plain-vanilla structural unit, 3.5.6), except that they cannot contain distributed-list items (3.5.5) or nested admonishments.

### 3.5.3 <Annote> (annotations)

The **`Annote`** element is used to enclose all types of annotations. The type of annotation is specified by the **`type`** attribute. These **`type`** values should be recognized: "Important"; "Attention"; "Note"; "FNote" (floating note/footnote); "AuthorNote"; "EditorNote"; "PublisherNote"; "Comment"; "ReviewerComment"; "ReaderComment". No distinction is made between footnotes, sidenotes, and endnotes: all "FNotes" (floating notes) will be treated as one of those types by the rendering application, according to its capabilities and the specifications provided by the user. (In an electronic environment there is no need to agonize over the placement of annotations: wherever they are, they're only a click away.)

Why is there one element for all kinds of annotations while there are four different kinds for admonishments? Some end users may have applications that do not look beyond the element name to decide how to format the element. It doesn't seem to matter much if an application can't figure out (from a **`type`** or **`label`** attribute) what kind of annotation it should format, but it may matter very much for admonishments, so we've removed any chance of not recognizing levels of importance in safety messages by giving each level its own element name.

### 3.5.4 Block lists (ordinary lists)

Four types of block lists may occur in a TIM document: unordered, ordered, variable, and segmented. Unordered lists have a mark, usually a bullet or dash, in front of each item. Ordered lists are numbered or lettered — there are many numbering options, described in Section 4.2. Segmented-list and variable-list items may be enumerated in all the ways that ordered-list items can be, but have special content models.

Every block list is divided into one or more sections, called groups, that contain the actual list items. This way of constructing lists allows subtitles to be inserted

in the list and allows changes in numbering properties, so that some items' numbers or marks can be emphasized or can be nonsequential. (Nesting of lists is a separate, fully supported feature.)

### 3.5.4.1 <UnorderedList>

An **UnorderedList** is a list in which each item is marked by the same character or string (such as a bullet or em dash). An **UnorderedList** begins with an optional (but unlikely) title and optional column headings (in a **ListHead** element), followed by any number of list items (**LI**s), which may or may not be grouped into **UnordListGrp**s. Multiple **UnordListGrp**s may be used within an unordered list to specify different marks or headings for parts of the list or individual items. An **UnordListGrp** may have its own title, which would be a subtitle for that portion of the list, and the whole list may have a title. Each **LI** may contain a single **P** paragraph or several para-like elements, including another whole list, with no limit on nesting.

Not everyone thinks of a bullet or dash list as "unordered," but when people think about the structure of their documents as much as SGML requires them to, they generally decide that the reason they sometimes use lists with bullets rather than numbers is that sometimes the order of the items really doesn't matter. Even so, as a rule SGML applications will respect the order you choose.

### 3.5.4.2 <OrderedList>

An **OrderedList** is a list in which the items are sequential, and therefore sequentially numbered or lettered. An **UnorderedList** begins with an optional (but unlikely) title and optional column headings (in a **ListHead** element), followed by any number of list items (**LI**s), which may or may not be grouped into **OrdListGrp**s. Several **OrdListGrp**s may be used within an ordered list to allow grouping of items with different headings or numbering properties.

The numbering styles that can be specified (for ordered lists, paragraphs, sections, and other things that can be numbered) are many: numerals, upper- and lowercase letters or roman numerals, or a symbol sequence (like *, †, ‡, §, **, ††, ‡‡, §§ for footnotes), with any prefix and suffix. Numbers of ancestor elements can be concatenated ("inherited"), as section numbers are in this document (e.g., 3.5.4.2). Numbers can also be "split," to give a sequence like 1, 2, 3a, 3b, 4, independently of any nesting of lists.

### 3.5.4.3 <VariableList>

A **VariableList** is a list in which "headwords," which might be glossary terms, error codes, or names, are followed by definitions or other explanatory text. A **VariableList** begins with an optional title and column headings (in a **ListHead** element), followed by any number of list items (**VarLI**s), which may or may not be grouped into **VarListGrp**s. Several **VarListGrp**s may be used within an ordered list to allow grouping of items with different headings or numbering properties.

A **VarLI** contains one or more **ListTerm**s (the "headword") paired with one or more **LI**s (the list-item body). There can be more than one of each, because two terms or messages may be paired with one definition, or one term might have multiple meanings.

#### 3.5.4.4 <SegmentedList>

A **SegmentedList** is a list in which each item (or row) has segments in a columnar alignment. The number of in-line items (or columns) is specified by the **segments** attribute.

Segmented lists correspond to certain constructs in some word processors (e.g., "snaking columns" in WordPerfect). For applications that do not have such constructs, segmented lists can be converted to tables with the same number of columns. On the other hand, tables with numbered rows may be better represented in TIM as segmented lists than as tables. Any list in a CALS table (CALS is the default table markup in TIM, 3.5.9) must be entirely enclosed in one table cell; no structure can span cells or rows of a table.

A **SegmentedList** begins with an optional title and optional column headings, followed by one or more required sections, called **SegListGrp**s. Several **SegListGrp**s may be used within a segmented list to allow different mark and list-item headings and/or numbering sequences within a single list. A **SegListGrp** contains one or more **SegLI**s, segmented-list items.

A **SegLI** is a "row" within a segmented list. It comprises one or more ordinary list items (**LI**s), which make up the columns of the row.

### 3.5.5 Distributed-list items

A distributed list is a list whose items are distributed throughout a document (or at least do not all occur within one parent element). Such lists, where items are numbered consecutively but do not appear contiguously, are commonly used in telecom documents for requirements, procedures, or steps of procedures. There are distributed versions of ordered and variable lists (a distributed unordered list seems impractical, since there would be very little clue that it was part of a list). They are both much like the correspondingly named block lists, except that there are no parent **...ListGrp** and **...List** elements. Numbering style and any headings over the parts of each item have to be respecified for each new item. Distributed-list items cannot occur inside paragraphs or each other. They exist only at the "base level" of document content, so that their relatedness is not further obscured by factors other than distance.

#### 3.5.5.1 <DistOrdLI>

A **DistOrdLI** is an item in a distributed ordered list. A **DistOrdLI** is not contained within a list element; instead, its **type** attribute should identify the list to which it belongs. At least the following **type** values should be recognized: "Procedure"; "Step".

A **DistOrdLI** has the same structure (content model) as an **LI** list item, except that that can be preceded by a title and column headings.

#### 3.5.5.2 <DistVarLI>

A **DistVarLI** is an item in a distributed variable list. A **DistVarLI** is not contained within a list element; instead, its **type** attribute should identify the list to which it belongs. The following **type** values should be recognized: "RqmtObj" (requirement object); "GlossEntry"; "AcronymEntry"; "Message".

A **DistVarLI** may have a **Title** and/or a **ListHead**, which provides headings over the "term" and "list item" columns of the list. Following the optional titles, it contains one or more **ListTerm**s followed by one or more **LI** list items. There can be more than one term, or more than one definition, in one variable-list item.

### 3.5.6 <S> (general-purpose structural grouping)

The **S** element is a general structural unit that can appear at any level of the **TelDoc** hierarchy, but is not itself part of the hierarchical structure. An **S** can be used to represent any kind of structure that isn't marked up more specifically (as a requirement object or a procedure would be). It is a flexible container element that can be used to group other elements together for any reason. An **S** cannot contain text directly; text must be contained within subelements — one of its uses is to wrap a sequence of subelements within an element that *can* contain text, to prevent line breaks and white space from being interpreted as content (3.8.3).

A **type** attribute can be used to indicate the purpose of the structural grouping (i.e., why a particular group of elements has been grouped into an **S** element). The **type** value of "Quote" should be recognized; however, because **S** can be used for virtually any reason, recipients should have a strategy for dealing with arbitrary **type** values.

A structural grouping may be titled or untitled; if titled, the **S** begins with a complex title (3.4.8). The optional title may be followed by any number of the following elements in any order: paragraphs (3.5.1), admonishments (3.5.2), annotations (3.5.3), block lists (3.5.4), distributed lists (3.5.5), code listings (3.5.11), and frames containing non-text elements (3.5.7). It can also contain other **S** elements, to create complex structural groupings.

Why is this structure called **S**? In a predecessor of TIM, **S** (for Section/Structure), **P** (Para), **LI** (List Item), and **T** (Text phrase) were almost the only elements used, and the names were kept short to keep the markup unobtrusive. As TIM evolved, many more-specific elements were added, and if the names had all been kept short, the meanings of most would be unrecognizable — as that of **S** is now.

### 3.5.7 <Frame>

The **Frame** is a holder for elements that aren't in the main text stream: figures, equations, code listings, or sidebar text. The **Frame** element corresponds fairly

well to the rectangular box that contains such elements on a printed page, although multipage content needs just one frame. The number and title of any enclosed figure or table should be considered part of the frame and not the element inside the frame. A **type** attribute should be used to differentiate the numbering streams for figures, tables, etc. The **type** values that should be recognized are "Table", "Graphic", "Figure", "Exhibit", "Icon" (untitled graphic)", "Equation", "Sidebar" (framed text), and "Listing".

A **Frame** has the same structure as an **S** structural grouping (3.5.6), except that it may also hold an **Equation** (3.5.8) or **GraphicalText** (3.6.3). **Graphic**s and **Object**s (3.5.10) will also most likely be in frames, and **Listing**s (3.5.11) may be.

### 3.5.8 <Equation>

An **Equation** is a mathematical equation or formula that may occur either in running text or within a **Frame** (3.5.7). If the **Equation** is numbered or titled, however, it must occur within a **Frame** (it's actually the frame that holds the number and title). The current release of TIM does not contain SGML markup specifically for equations (a modular enhancement is planned). For now, an **Equation** is likely to contain a graphic or a non-SGML encoding of the equation (e.g., TeX).

### 3.5.9 <Table> and <TGroup>

The **Table** element conforms to the CALS model for tables. The **Table** comprises one or more table groups, called **TGroup**s. The **TGroup** defines the default specifications of a table or portion of a table. The **TGroup** element has several attributes that define the appearance/formatting of the table:

- **align**: specifies the alignment of text within columns
- **char**: specifies special character alignment within columns
- **charoff**: specifies character offset within columns
- **cols**: specifies the number of columns in the **TGroup**
- **colsep**: specifies the vertical ruling between columns
- **rowsep**: specifies the horizontal ruling between rows
- **tgroupstyle**: specifies a predefined style name for the **TGroup**

The **TGroup** begins with optional column specifications (3.5.9.1) and span specifications (3.5.9.2). After the optional specifications, a **TGroup** may contain an optional table header (3.5.9.3), an optional table footer (3.5.9.4), and a required table body (3.5.9.5).

#### 3.5.9.1 <ColSpec> (column specification)

One **ColSpec** should be defined for each column in a table group. The **ColSpec** element contains no data; instead, it defines the specifications for a given

column through a series of attributes: **align**, **char**, **charoff**, **colsep**, and **rowsep** can be used to override the **TGroup** default settings. There are these additional attributes:

- **colname**: specifies a name for a column
- **colnum**: specifies sequential number of column from left to right
- **colwidth**: specifies width of a column

### 3.5.9.2 <SpanSpec> (span specification)

The **SpanSpec** is used to specify which columns will be spanned by cells in the rows within the **TGroup**. The **SpanSpec** element contains no data; instead, it defines the specifications for a given span through a series of attributes: **align**, **char**, **charoff**, **colsep**, and **rowsep** can be used to override the **TGroup** or **ColSpec** default settings. There are these additional attributes:

- **namest**: names the first column a cell should span
- **nameend**: names the last column a cell should span
- **spanname**: specifies a name for the column spanning specification

### 3.5.9.3 <THead> (table header)

A **THead** contains the header rows for a **TGroup**. The **THead** element has a **valign** attribute that specifies the vertical alignment of text within the header rows. The **THead** may begin with zero or more **ColSpec**s to change the default values inherited from the parent **TGroup**. After the optional specifications, the **THead** contains one or more rows of headings.

### 3.5.9.4 <TFoot> (table footer)

A **TFoot** is a footer for a **TGroup**. The **TFoot** element has a **valign** attribute that specifies the vertical alignment of text within the footer rows. The **TFoot** has the same structure as a **THead** (3.5.9.3). It precedes the body of the table in the SGML file so that its contents are available for each page of a paginated display (this is a concession to applications that cannot look ahead when formatting a file).

### 3.5.9.5 <TBody> (table body)

The **TBody** is the element that contains the body rows of a **TGroup**. Only one **TBody** may occur within a **TGroup**.

### 3.5.9.6 <Row> (table row)

A **Row** contains one or more table cells (**Entry**) or nested tables (**EntryTbl**). A **Row** should contain one **Entry** or **EntryTbl** for each column in the table, unless a **SpanSpec** has been used to specify that some cells are combined.

#### 3.5.9.7 <Entry> (table cell)

An **Entry** is the content of a table cell. There should be one **Entry** for each column in the table, unless a **SpanSpec** is used. In TIM, **Entry** has the same structure as an **S** structural grouping (3.5.6).

#### 3.5.9.8 <EntryTbl> (nested table)

An **EntryTbl** is a subtable within a table cell. An **EntryTbl** may be substituted for an **Entry** in a row. The **EntryTbl** has the same structure as the **TGroup** element, except that is cannot contain a **TFoot** or another **EntryTbl**.

### 3.5.10 <Graphic>, <Object>, <Flowchart>, and <ExternalText>

An SGML *file* may contain only ASCII characters. This does not prevent an SGML *document* from containing graphics, sound clips, or multimedia extravaganzas. They simply must be in separate files. The elements **Graphic**, **Object**, **Flowchart**, and **ExternalText** invoke external files with content that does not fit into the main SGML file. Graphics are 2-dimensional images that can be represented on a static page or screen. Objects are things that require special rendering: 3-D or moving images, sounds, or some combination of those. External text may be text with word-processor codes or just something that could be in the SGML file but, for some reason, is stored separately.

External entity references in a TIM document are made from a **Graphic**, **ExternalText**, **Object**, or **Flowchart** element, like this:

```
<Graphic entityref="fig001">
```

Or, for **NmList**:

```
<NmList docorsub="Tedd">TCIF-IPI-95-001-XREF-342</NmList>
```

When an entity reference is used, there must be a corresponding entity declaration in the prologue of the document instance, relating the entity to an external file:

```
<!DOCTYPE TDoc PUBLIC "-//USA/TCIF//DTD TIM-2//EN" [
<!ENTITY fig001 SYSTEM "../tiff/fig001.tif" NDATA TIFF>
<!ENTITY Tedd SYSTEM "../../../ipi95001/mtext/main.tim" NDATA
SGML>
...
]>
<TDoc>
...
```

In the entity declaration, "SYSTEM" means that the identifier that follows should be interpreted directly by the system rather than looked up in a catalog of public identifiers (which is how "-//USA/TCIF/DTD TIM-2//EN" is to be interpreted).

Because the TEDD packaging guidelines (TCIF-IPI-95-001) say exactly where such files should be, these system identifiers should reliably locate the right files.

"NDATA" says that the file contains non-SGML data, and the next term says what kind. (Since "NDATA" is required by the SGML standard in an entity reference, even a TIM file must be described as "non-SGML data.") The data-type information can be used to select an application appropriate for presenting the data — in the case of "TIFF", the file should be opened by a TIFF image viewer. TIFF is one of over 30 NOTATIONs (data formats) already defined in TIM.

### 3.5.11 <Listing> (program code or other literal text)

Sometimes it is necessary to display a program code listing or other text that requires preservation of white space and line breaks, and may also require non-proportional text. The **Listing** element can be used to display such "literal" text. A **Listing** may have several lines of code within it, or each line of code may be enclosed in a separate **Listing** element (allowing individual lines to be numbered).

The rendering application should try to preserve the line breaks it finds in a listing (some applications may ignore them, see 3.8.3). A "&newline;" entity can be inserted if necessary to make line breaks explicit. In case the lines are too long, the **width** attribute can be used to let applications or users know the number of characters that were intended to fit on one line in the original representation of the data (e.g. **width="80"**). The **type** attribute can be used to indicate the type of data contained in the **Listing**. At least the following **type** values should be recognized: "Example"; "Code"; "Input"; "Output"; "ScreenImage".

## 3.6 Text-level elements (substructure)

Text-level elements are wrappers that are placed around a string of text in order to provide it with a special meaning or presentation.

### 3.6.1 <Sub> and <Sup> (subscripts and superscripts)

**Sub** indicates that characters should appear below the baseline (a subscript). **Sup** indicates that characters should appear above the baseline (a superscript). The actual formatting of superscripts and subscripts depends on the output system. **Sub** and **Sup** can occur anywhere in text, and can be nested.

### 3.6.2 <Symbol>

The **Symbol** element marks characters in non-alphabetic fonts.

### 3.6.3 <GraphicalText>

**GraphicalText** contains text characters that draw a picture, like `:-)`. **GraphicalText** may occur either in running text or within a **Frame** (3.5.7). It is different from literal text (**Listing**, 3.5.11), which is still literal if presented

in proportional fonts. Graphical text conveys its message only if its shape is preserved. Fonts are not specified anywhere in SGML, but rendering applications can assign fonts to elements. Most special characters, like box-drawing characters, can be represented by their entity names.

### 3.6.4 <T> (text phrase)

A document can have phrases within running text that have special meaning or require specific presentation (e.g., trademarks, document titles, quotations, filenames, and so on). The **T** element can be used to capture special types of text phrases. The type of phrase that the **T** element represents is specified by the **type** attribute. **T** elements can be nested to create complex text phrases.

**T** is for text phrases that are marked for semantic reasons — that is, they're something someone might be looking for, like error codes, trademarks, or part numbers. If a word or phrase is made typographically distinct just so it will stand out, use **Emph** instead of **T**.

### 3.6.5 <Emph> (emphasis)

The **Emph** element is used to place stylistic emphasis on text. The type of emphasis is specified by the **emph** attribute. The following **emph** values should be recognized: "ital" (*italic*); "bold" (**boldface**); "boldital" (***boldface italic***); "roman" (plain); "und" (underline); "dblund" (double-underline); "color" (color text); and "sys" (SYSTEM-SPECIFIC).

### 3.6.6 <SimpleList>

A **SimpleList** is a simple list, one with uncomplicated items that could be placed in a comma-separated series in one sentence, in a column, or in several columns.

### 3.6.7 <IndexTerm>

An **IndexTerm** occurs in text to provide the raw material for an **IndexEntry** (3.7.2.5). The content of the **IndexTerm** should not be renderedwhere it occurs.

## 3.7 Metastructure

A TIM document contains several elements that can be used as navigational tools to help traverse the data of a document or set of documents. The following navigational tools are available in TIM documents: alternative representations, cross-references, floating elements, HyTime location addresses, links, navigational entries, and navigational lists.

### 3.7.1 <Pt> (alternative and virtual structures)

**Pt** is a dimensionless point that can be used to mark the start or end of a virtual structure. It is allowed to appear almost anywhere inside any other element, so it provides flexibility that SGML's structural rules might otherwise prevent. SGML

insists that structure be strictly hierarchical: each element must be completely within its parent. So an **Emph** element indicating italics could not start in one paragraph and end in another. And elements could not be placed in an SGML file to divide a document into pages, if it is already divided into paragraphs that don't always start and end on the same page. But the alternative structure can be represented by marking start and end points with empty elements that do not risk crossing existing structural boundaries.

## 3.7.2 Cross-references and links

A cross-reference is a pointer from one element or piece of information to another (e.g., "see Section 4"). Two different elements are used for cross-references in TIM documents:

### 3.7.2.1 <Ref>

**Ref** is a simple in-line reference in text to such things as footnotes, tables, and bibliographic citations. **Ref** is a wrapper that can be placed around text, numbers, or marks to create a cross-reference. A **Ref** can be used to reference elements within the document or, using special HyTime constructs described below, elements in other documents. A **Ref** must have an **rid** attribute that matches the ID of the target element (for internal references), or of a **NameLoc** (for external references, 3.7.2.3.1). Authors should expect that the text content of a **Ref** may be altered or discarded by end users, since a textual reference that works on paper (e.g., "see page 3-21") may not work when the document is viewed on line in a pageless presentation and the reference has become a hyperlink.

### 3.7.2.2 <URLRef>

**URLRef** is a simple reference to a Uniform Resource Locator (a World Wide Web address). The **url** attribute is used to specify the URL to be used for hyperlinking. Document browsers that can interpret these references can connect to the referenced location.

### 3.7.2.3 HyTime location addresses

HyTime is an extension to SGML that provides a scheme for creating links to external elements. HyTime provides several ways to address the location of an element that is outside of the current working document. To create an external cross-reference, the **Ref** (3.7.2.1) points to the "location address" of the element (instead of the element itself, as is done for internal references). For location addressing to work, there must be an external entity declaration in the document prologue for each external document containing target elements.

HyTime supports three methods of addressing external elements: named location addresses (**NameLoc**), tree location addresses (**TreeLoc**), and data location addresses (**DataLoc**).

#### *3.7.2.3.1 <NameLoc> (named location address)*

The **NameLoc** is used when the external element has been "named" with an **ID** attribute value. To reference an external element that has been named, a **Ref** is used to point to a corresponding **NameLoc**; which in turn points to the external source document and the ID of the element within that document. A NameLoc is made up of one or more name specifications lists (**NmLists**), which specify the name and location of an external element.

The **NmList** specifies the location of the named object using the following attributes:

- **nametype**. This attribute indicates the type of object that is being named. It can have the following values: "element" (an element in the target document); "entity" (the entire target document).
- **docorsub**. This attribute indicates the name of the external entity (document) that the target object is in.

Because the **NameLoc** links an external element by its ID, it creates a robust link that is not affected by structural changes in the target document.

#### *3.7.2.3.2 <TreeLoc> (tree location address)*

If the intended target of a cross-reference does not have an **ID** attribute value assigned to it, its location can be specified according to its nesting level in the tree structure of the document. The **TreeLoc** tracks an element's nesting path in the tree structure through a series of markers. The **locsrc** attribute is used to specify the **ID** of the top-level element in the tree (for internal links), or the **ID** of a **NameLoc** (for external links).

Because the **TreeLoc** links an external element by its place in the document structure, it creates a fragile link that can be broken if there are any structural changes in the target document. Whenever possible, **NameLoc** should be used instead.

#### *3.7.2.3.3 <DataLoc> (data location address)*

Sometimes the intended target of a cross-reference may be a stream of data within an element (and not the element itself). The **DataLoc** can be used to identify the start and end points of a data stream within an element through one or more pairs of dimension specifications. The **locsrc** attribute is used to identify the element in which the data stream is to be counted (for internal links), or the **ID** of a **NameLoc** or **TreeLoc** (for external links). The **quantum** attribute specifies the type of data tokens to be counted (e.g., words or characters).

Because the **DataLoc** links a data stream by counting its start and end points, it creates a fragile link that can be broken by any changes to the text of the target document; therefore, **DataLoc** should only be used to link to data streams in static documents..

#### 3.7.2.4 Links

Links are empty elements within a document that establish links to other internal or external points. Unlike cross-references, which are text phrases that point to other data, link elements have no defined content (i.e., they are empty), they merely define a connection between two distinct physical points in a document or set of documents. Links may appear virtually anywhere in a **TDoc** instance.

#### 3.7.2.5 <IndexEntry>

**IndexEntry** may appear in an already formatted **Contents** or **Index** element. **IndexEntry**s can also be generated in later processing from **IndexTerm**s embedded in the text. For nested entries, **IndexEntry** can be nested. A **Ref** subelement provides each reference to places in the text.

#### 3.7.2.6 <Contents> and <Index> (navigational lists)

Contents and index lists are navigational lists that point to the location of other elements within the document. The **Contents** element can be used for items such as a table of contents, list of figures, list of tables, etc. The **Index** element can be used for alphanumerically sorted lists of contents such as a typical index or an alphabetized list of some other specified element. Both **Contents** and **Index** can occur at any level of **Section** (3.4.7).

## 3.8 Formatting a TIM file

A TIM file (formally, a TIM document instance) contains more than just a nesting of elements. There are usually some lines at the top of the file that aren't elements. And then among the elements there may be various strange character sequences of the form "&name;". There may also be inexplicable patterns of white space, line breaks, even blank lines.

### 3.8.1 The document prologue

A proper TIM document instance begins with a DOCTYPE declaration:

```
<!DOCTYPE TDoc PUBLIC "-//USA/TCIF//DTD TIM-2//EN">
```

This declaration says the all-encompassing element is "TDoc", and that TDoc and all its subelements are defined in the file(s) located using the public identifier in quotes. Public identifiers follow certain rules, and are meant to be looked up in a catalog that your SGML applications are aware of, so that they can find the file(s) corresponding to that identifier.

The DOCTYPE declaration will usually be supplemented like this:

```
<!DOCTYPE TDoc PUBLIC "-//USA/TCIF//DTD TIM-2//EN" [
<!ENTITY fig001 SYSTEM "../gif/fig001.gif" NDATA GIF>
<!ENTITY fig002 SYSTEM "../gif/fig002.gif" NDATA GIF>
<!ENTITY fig003 SYSTEM "../gif/fig003.gif" NDATA GIF>
```

```
]>
```

Three things can occur between the square brackets:

- Declarations of external entities (files) containing graphics and any other non-SGML (non-ASCII) data. The declarations above for "fig001", etc., are of this type. Every external file that is part of the document must be named in the prologue. Then it is referred to by its entity name where it occurs in the document:

  ```
  <Graphic entityref="fig001">
  ```

- Extensions of the DTD (several are planned; others are not to be encouraged, but at some point they may be necessary). There may be element or attribute declarations, or declarations for entities used within existing element/attribute declarations. For instance, the TIM DTD's declaration for the **`Equation`** element could be replaced with one that includes new subelements:

  ```
  <!ELEMENT Equation - - (Oper|Var|Constant|Function)*>
  ```

  The four new elements would also be defined there. The new declaration for **`Equation`** would replace the one in the DTD: any declaration in the prologue takes precedence over a declaration for the same element or entity in the DTD.

- Comments. This is a comment:

  ```
  <!--This is a comment.-->
  ```

### 3.8.2 Entity references

Since SGML files contain only ASCII characters, they have to represent non-ASCII characters like ∞, Σ, ♥, ≠, and e using ASCII characters. Each of more than 600 special characters routinely defined for TIM has a name that starts with "&" and ends with ";". For instance, a bullet is "•" and a Sigma is "Σ" (case matters in entity names: "Σ" and "σ" are different entities). The character entities are defined in eight separate files, "entity sets," seven with ISO standard characters and one with TCIF characters representing keyboard keys.

### 3.8.3 Line breaks and white space

Line breaks and extra spaces are both used in arbitrary ways in the TIM instance example (Exhibit 3-1). SGML treats them this way:

1. If the current element has only subelements in its content model, not text (which is known formally as PCDATA, "parsed character data"), then spaces and line breaks are ignored. **`TDoc`** is the current element at the beginning of the third line of the example, so the spaces in front of "**`<DocID>`**" are meaningless, there just for readability.
2. If an element is allowed to have text in it directly, then interword spaces are part of the content. Line breaks in running text are treated as spaces — it's

assumed that line breaks will be recalculated by whatever application is formatting the SGML file for presentation. Two or more consecutive spaces and/or line breaks are collapsed into a single space. If you still type two spaces between sentences, you might as well stop. And if you hit e an extra time to get extra space between paragraphs, learn how to adjust the "Space Above" property for particular paragraph styles in your word processor.

Tab characters are also converted to collapsible spaces, because it is hopeless to try to get different systems to do the same thing with them (this is a practical problem, not a failing of SGML). *All tabs should be banished from documents to be translated to TIM.* If you start paragraphs with a tab, specify an automatically indented style instead — the recipient can choose that style or not for displaying the paragraphs. If you format columnar lists with tabs, find out how your word processor can do it automatically, and use "segmented lists" or tables in TIM.

3. If an element has a NOTATION (data format other than TIM) defined for it, using the **`format`** attribute, then line breaks, spaces, tabs, and all other legal ASCII characters will be preserved. Some applications also allow elements to be defined as "verbatim" without using a notation, so white space and line breaks can be preserved while TIM markup characters are interpreted. This is the assumed default for the elements **`Listing`** and **`GraphicalText`**.

# 4 Attributes

Attributes are characteristics or values that can be assigned to elements to provide additional information about the structure they represent or the data they contain. There are several different types of attributes in TIM:

1. **Descriptive attributes**. These attributes describe the purpose or function of an element at a particular place in the document.
2. **Enumeration/labeling attributes**. These attributes specify any automatically generated label that a rendering application should apply to an element (typically, numbering or a fixed label like "CAUTION!").
3. **Display-oriented attributes**. These attributes provide specifications for how an element is to be displayed by a rendering application. (There are only a few of these).
4. **Cross-reference attributes**. These attributes provide information used to create cross-references or other links between elements.
5. **Navigational attributes**. These attributes are used to generate navigational aids, such as **`Contents`** and **`Index`** lists.
6. **CALS table attributes**. These attributes are used to specify the display characteristics of a table.

Some attributes can be given almost any text value:

```
type NMTOKENS #IMPLIED

dstype CDATA #IMPLIED

target NMTOKEN #IMPLIED
```

A NMTOKEN is a string containing only numbers, letters, "." or "-". CDATA is any sequence of ASCII characters, including spaces. Other attributes, declared like these, can take only a value from the list (separated by "|"):

```
emph (ital|bold|boldital|roman|und|dblund|color|sys) #IMPLIED

align (Left|Right|Center|Justify|Char) "Left"
```

In the example, **`align`** has a default value, "Left". The other attributes, like most in TIM, have an "implied" value, which is more flexible than an explicit default, as long as the processing system is aware of the implied defaults. They are described in TCIF-IPI-95-004-PART2.

**NOTE:** Attribute names are normally shown in lowercase, but will be capitalized at the beginning of a sentence. Case is irrelevant in processing. Attribute values are shown in quotes, and in this document closing quotes for values are always placed before commas and periods to prevent misinterpretation.

## 4.1 Descriptive attributes

### 4.1.1 **type** and **dstype**

**Type** is used to indicate the primary semantic type of an element. It describes an element's specific role, which may be different at different places in the document. For example, **Section** elements subdivide the body of a document; that is their structural role. However, it is the **Section** element's **type** attribute that determines the *purpose* of the subdivision (chapter, glossary, appendix, etc.). Thus a bibliography section would have the following markup:

```
<Section type="Bibliography">...</Section>
```

The **type** value should be a NMTOKEN (single word with no white space). There are no predefined values for **type**; however, the TCIF IPI Committee has agreed on certain common **type** values that should be recognized for each of several elements. Refer to an element's definition in TCIF-IPI-95-004-PART2 for any list of its recommended **type** values.

**Dstype** can be used to specify a subtype of a category defined by the type attribute. For example, the markup for a program code listing might be:

```
<Listing type="Code" dstype="c++">...</Listing>
```

### 4.1.2 **group**

**Group** is used to indicate an element group (or groups) to which a particular element belongs. Unlike **type**, **group** does not imply anything about an element's purpose, structure, or formatting. It is an arbitrary grouping that can be used to classify elements for special purposes, such as inclusion in an index, table of contents, etc.

### 4.1.3 **remap**, **remapatt**, and **remapto**

**Remap** is used to indicate how to map a particular element when converting to a different markup. For example, the glossary section of a TIM document that was converted from the DocBook DTD, which has a **Glossary** element, might have the following markup (if conversion back to DocBook is anticipated):

```
<Section type="Glossary" remap="Glossary" remapto=”DocBook”>
... </Section>
```

The **remap** attribute is for conversion mapping only; it should not be used to specify the function of an element within the current document, as **type** is.

Sometimes there could be attributes in the original markup that do not exist in TIM. They can be preserved with the remapatt attribute.

### 4.1.4 **revstat**

**Revstat** is used to indicate whether an element's contents have been changed, and whether the change should be indicated to the reader. For instance, a value

of "revised" indicates that the element has been revised, and "revflag" indicates that the revised element should be flagged as such (e.g., with a changebar).

### 4.1.5 status

**`Status`** can be used to indicate various administrative information about an element. It can be used to differentiate between concurrent versions of an element:

```
<Section status="v8.0.3">...</Section>
```

It can be used to indicate the security access or clearance of an element:

```
<P status="proprietary-internal">...</P>
```

It can be used to indicate the effectivity/applicability of an element:

```
<Section status="BellSouth, Ameritech">...</Section>
```

### 4.1.6 lang

This attribute is used to specify the natural language of an element's text content. The possible values of **`lang`** are ISO-639 language codes.

### 4.1.7 keywords

**`Keywords`** is used to specify any keywords that the author considers useful for cross-referencing or categorizing an element.

### 4.1.8 format

This attribute is used to specify the data representation used for an element's contents, if other than TIM markup. This attribute can be used for elements that contain literal text or non-SGML data.

The **`format`** attribute must have a value that is a NOTATION (a format whose processing requirements have been formally defined within the rendering application or an external helper application). There are several predefined values for the format attribute, such as:

- "RTF". Indicates that the element's content is encoded in Microsoft Rich Text Format.
- "HTML". Indicates that the element's content is encoded in Hypertext Markup Language.
- "SGML". Indicates that the element's content is encoded in SGML.
- "TEX". Indicates that the element's content is encoded in TeX.
- "TBL", "EQN", "PIC". Indicates that an element's content is a **troff** TBL table, EQN equation, or PIC graphic.

## 4.2 Enumeration/labeling

The TIM attributes below can be used in combination to specify numbering algorithms at least as complex as those that current end-user applications can support.

Numbers and other labels on elements are not considered content; they're just a convenience for navigating through the items and knowing where you are. The end user's application may not be able — or the end user may not want — to number items the same way. So these specifications will be treated as suggestions only.

### 4.2.1 **mark** and **numeration**

**`Mark`** specifies the mark (e.g., bullet or em dash) that appears before list items in an unordered list. The **`mark`** value can be an entity reference to a non-keyboard character. For example, a bulleted list would have the following markup:

```
<UnordListGrp mark="&bull;">...
```

**`Numeration`** specifies the type of numbering/lettering for a sequence of numbered elements (such as sections, numbered paragraphs, or ordered-list items). For example, a value of "arabic" specifies that elements should be numbered (1, 2, 3 ...). The other choices are "upperalpha", "loweralpha", "upperroman", "lowerroman", "outline" and "symbol".

```
<OrdListGrp numeration="arabic">...</OrdListGrp>
```

If the choice is "symbol", **`symbolseq`** (Section 4.2.8) specifies the sequence of symbols.

### 4.2.2 **prefix** and **suffix**

**`Prefix`** and **`suffix`** specify an optional prefix and suffix for the number/letter/symbol of an enumerated element:

```
<Section type="Appendix" numeration=upperalpha
prefix="Appendix " suffix="." label="Appendix A."> ...
</Section>
```

**`Label`** shows the overall result of the specified prefix, enumeration, and suffix.

### 4.2.3 label

**`Label`** specifies an element's complete number or mark in case it cannot be calculated automatically by the rendering application. For example, if the rendering application cannot automatically construct the proper number for a Figure, the **`label`** attribute can be used to explicitly specify the correct label:

```
<Frame type="Figure" numeration=arabic inheritnum=inherit1
inheritfrom="Section" prefix="Figure " suffix="." infix="-"
label="Figure 1-1."> ... </Frame>
```

It is highly recommended that labels always be supplied for numbered elements, because the autonumbering capabilities of some rendering applications are certainly less than TIM assumes. The autonumbering building-block attributes are helpful when elements are rearranged and new numbers must be calculated, but to render a document as delivered, it is much easier just to display the supplied label.

### 4.2.4 **inheritnum**, **inheritfrom**, and **infix**

**`Inheritnum`** specifies whether a numbered element should have the numbers of its parent(s) concatenated with its own number. For example, Section 4.2.4 inherits its number from its parent, Section 4.2, which, in turn, inherits its number from Section 4.

The **`inheritnum`** attribute has several predefined values that determine the level of ancestor from which the number is inherited.

**`Inheritfrom`** specifies the element(s) from which numbers are to be inherited when **`inheritnum`** is specified. For example, a table contained in a third-level subsection may inherit the number of its first-level section ancestor (e.g., Table 3-1 instead of Table 3.2.1-1).

```
<Frame type="Table" inheritnum="inherit1"
inheritfrom="Section" infix="-" ...>
```

**`Infix`** specifies a separator for concatenated numbers, such as a period ("Section 1.2") or a hyphen ("Figure 1-2").

### 4.2.5 **continuation** and **restartsat**

**`Continuation`** determines whether the number of a numbered element (such as a section, numbered paragraph, or ordered list item) should increment or restart at 1, or whatever value is specified in **`restartsat`**.

The **`continuation`** attribute has three predefined values:

- "continues": element number increments based on the previous number in the sequence. This is the typical default for most numbered elements (e.g., Section 1, Section 2, Section 3, etc.).
- "restarts": element number restarts at 1, or at another number specified by the **`restartsat`** attribute. This is the typical default for nested lists. Example:

  A. List item

     1. List item

     2. List item

  B. List item

     1. List item

     2. List item

The value of **restartsat** is a number. If this attribute is not defined, its value is assumed to be 1.

- "holds": element number retains the same value as previous element number. This is typically used for split numbering (Section 4.2.6).

### 4.2.6 splitnum

**Splitnum** is used to specify whether a number in a series of numbered elements (such as sections, numbered paragraphs, or ordered list items) should be split among two or more items (e.g., 1, 2, 3a, 3b, 3c, 4, 5...).

### 4.2.7 increment and shownum

**Increment** is used to specify the numbering increment between successive items numbered in the same sequence: "-1" would indicate numbering like 5, 4, 3, 2, 1. **Shownum** is used to specify on which items the number is visible. Each number count off items to the next one numbered, and the last number is reused: "1 2 3" means show a number on the 1st, 3rd, 6th, 9th, etc. items in a sequence.

### 4.2.8 symbolseq

This attribute is used to specify the sequence of symbols to be used if **numeration** is set to "symbol". For example, footnotes might be marked using "*, †, ‡, §" instead of "1, 2, 3, 4."

## 4.3 Display-oriented attributes

Only the display-oriented attributes available for most elements are covered here. There are also attributes that apply only to **SimpleLists**, **SegmentedLists**, **Listings**, **Frames**, **Graphics**, and **Objects**.

### 4.3.1 present

The **present** attribute specifies when and where an element's contents should be presented. The default, of course, is "normal", meaning present the content where you find it in the data stream. The alternatives are "float", present it in addition to the element that references it (probably in a user-controlled pop-up window) when the user requests it (probably by clicking on a hotspot); "alert", present it in addition to the element that references it without waiting for the user to request it; and "altrep", present it in place of the element that references it when the user requests it.

### 4.3.2 emph

This attribute is used to specify the stylistic emphasis that is to be placed on the contents of an element by a rendering application (such as boldface, underscore, etc.).

The **emph** attribute has several predefined values: "ital" (italic), "bold" (boldface), etc. Rendering applications should be capable of processing each of

the values; however, there is no guarantee that the they will be able to do exactly what the attribute value calls for.

Even the **Emph** element uses an **emph** attribute to specify the type of emphasis intended; if no **emph** value is specified, the implied value is "ital". For all other elements, no specified **emph** value means that no emphasis is intended.

### 4.3.3 presspec

**Presspec** attribute is used to provide any presentation specifications for an element other than emphasis. There are no predefined values; however, a value of "centered" should be recognized.

## 4.4 Cross-reference attributes

All the attributes listed in this section are used to create cross-references or other kinds of links. Attributes used only within HyTime elements are covered only in TCIF-IPI-95-004-PART2.

### 4.4.1 id

This attribute specifies a unique identifier (sometimes called a "name") for an element. This attribute must be assigned to any element that is to be the target of any type of link. Since it is not always possible to determine in advance whether a link will point to an element, it is good practice to assign IDs to all appropriate elements whenever possible.

The **id** value is an ID (unique single word with no white space) with a maximum length of 64 characters.

### 4.4.2 rid

**Rid** is used by a **Ref** element, the usual internal or HyTime-based cross-reference, to identify the object it points to. The **rid** value must match the **id** value of the target object. For example, if a **Ref** points to an element with an **id** value of "TCIF-IPI-95-004-SECTION-1," then the following markup would be used:

```
<Ref rid="TCIF-IPI-95-004-SECTION-1">...</Ref>
```

The value of **rid** is defined to be an IDREF, a reference to an ID. SGML parsers are supposed to check to make sure that each IDREF matches an ID in the same document.

### 4.4.3 alerts

The **alerts** attribute is used to create a cross-reference to one or more elements that should appear at the same time as the referencing element. The **alerts** value is a list of the **id** values of the elements being referenced. Multiple ID values should be separated by white space.

### 4.4.4 altreps

The **altreps** attribute is used as a pointer to one or more alternative representations of an element's content. The **altreps** value is a list of the **id** values of the alternative elements being referenced. Each **id** value should be separated by white space.

### 4.4.5 spanend

**Spanend** is used by **Pt** to point to the element that is the endpoint of a virtual structure. The **spanend** value must match the **id** of the element being pointed to.

### 4.4.6 linkends

**Linkends** is used to provide exactly two ID references for the elements at the ends of a **Link**. For example, if a **Link** element points to one element with an **id** value of "element1" and another with an **id** value of "element2," then the following markup would be used:

```
<Link linkends="element1 element2">
```

## 4.5 Navigational-element attributes

These attributes allow specification of the content to be generated for textual cross-references, for tables of contents, and for indexes. Ready-made content can also be provided, but it will be obsolete if any changes are made to the document. (If it includes page numbers, it can be obsolete even without changes.) It is understood that the specifications provided are suggestions, not requirements. Recipients may not have applications that can follow the specs, they may have applications that can do even more, or they may just prefer something different than what is specified for the sake of consistency.

### 4.5.1 target

**Target** may be used to specify the type of element that is the target of a **Ref** or **URLRef**. The value of target is the name(s) of the target element(s). **Target** is helpful in generating wording for a cross-reference when the original wording is inappropriate (e.g., "see page 7").

### 4.5.2 include

**Include** may be used to specify the kinds of content to be generated for a **Contents** list, **Index**, **Ref** or **URLRef**. There are no predefined values for **include**; however, the following values should be recognized: "num" (element number or mark), "title" (element title), "text" (element text), "allcontent" (element content of all forms), "page" (page number), and "link" (hypertext link from contents/index list item to element). Like **target**, **include** is helpful in generating wording for a cross-reference when the original wording is inappropriate.

### 4.5.3 elements and levels

**Elements** is used to specify what elements should be listed in a **Contents** list or **Index**. For example, a **Contents** list could specify a Table of Contents this way:

```
<Contents elements="Section">...</Section>
```

Like **target**, the **elements** attribute can have a value of one or more NMTOKENS (single words separated by white space). The default value of this attribute should be understood to be "Section" for **Contents** and "IndexEntry" for **Index**.

**Levels** is used to specify what nested levels of an element should be listed. For example, a Table of Contents that displays only 1st- and 2nd-level sections could be marked up as:

```
<Contents elements="section" levels="1 2">...</Contents>
```

The **levels** value is one or more numbers separated by white space.

### 4.5.4 types

**Types** is used to specify what values of **type** and/or **dstype** (Section 4.1.1) an element must have in order to be included in a **Contents** list or **Index**. For example, a **Contents** list could specify a List of Figures this way:

```
<Contents elements="Frame" type="Figure">...</Contents>
```

### 4.5.5 groups

**Groups** is used to specify what value(s) of **group** (Section 4.1.2) elements must have in order to be included in a **Contents** list or **Index**.

### 4.5.6 remaps

**Remaps** is used to specify what value(s) of **remap** (Section 4.1.3) elements must have in order to be included in a **Contents** list or **Index**.

### 4.5.7 revstats

**Revstats** is used to specify what value(s) of **revstat** (Section 4.1.4) elements must have in order to be included in a **Contents** list or **Index**.

### 4.5.8 statuses

**Statuses** is used to specify what value(s) of **status** (Section 4.1.5) elements must have in order to be included in a **Contents** list or **Index**.

### 4.5.9 langs

**Langs** is used to specify what value(s) of **lang** (Section 4.1.6) elements must have in order to be included in a **Contents** list or **Index**.

### 4.5.10 scope

**Scope** is used to specify the parent elements that should be searched for subelements that meet the criteria specified by the **Contents** list or **Index** (**elements**, **types**, **levels**, etc.).

The **scope** value is the name(s) of the element(s) to be searched. If this attribute is left undefined, then the whole document is to be searched. The special value "Parent" means the one parent element enclosing the **Contents** or **Index** element. The contents of a **Contents** or **Index** should not be searched.

### 4.5.11 sortorder

**Sortorder** is used to specify where nonalphabetical characters in an **Index** should sort relative to alphabetical characters and each other.

### 4.5.12 sortas

**Sortas** is an attribute of all elements that can contain text. It is used to specify how to alphabetize the element's content in an **Index**: to ignore symbols, spell out numerals, etc. For example, suppose that "#" were an index entry that was supposed to be alphabetized as though it were spelled out (e.g., "pound sign"). The following markup would be used:

```
<IndexTerm sortas="pound sign">#</IndexTerm>
```

In an index list, the entry would appear here:

```
POTS

#

Power Supplies
```

The value of **sortas** can be any string of characters including white space. There are no predefined values.

## 4.6 CALS table attributes

### 4.6.1 orient

**Orient** is used to specify whether the table is intended to be in portrait or landscape orientation on an ordinary paper page. The value of **orient** is either "port" (portrait) or "land" (landscape). If **orient** is undefined, its value is assumed to be "port".

### 4.6.2 pgwide

**Pgwide** is used to specify whether a table should be expanded to the full text width of a page, rather than stay within the margins of its text column (for multicolumn formats). A **pgwide** value of "1" (yes) indicates that the table should span the entire page; a value of "0" (no, the default) indicates that the table should remain within the margins of its text column.

### 4.6.3 frame

The **frame** attribute specifies whether a border is to appear around the outside of the table. It has the following predefined values: "top" (border to appear only on top of table), "bottom" (border to appear only on bottom of table), "topbot" (border to appear on top and bottom of table), "sides" (border to appear on right and left sides of table), "all" (border to appear on all four sides of table), and "none" (no border). If frame is undefined, the rendering application must choose a default.

### 4.6.4 colsep

**Colsep** indicates the type of separator to be used between the columns of a table. The value of **colsep** is a number. A value of "1" indicates a vertical rule between columns, a value of "0" indicates that there is no separator.

### 4.6.5 rowsep

**Rowsep** indicates the type of separator to be used between the rows of a table. The value of **rowsep** is a number. A value of "1" indicates a horizontal rule between rows, a value of "0" indicates that there is no separator.

### 4.6.6 cols

**Cols** is used to specify the number of columns in a **TGroup** or **EntryTbl**. There is no default value for **cols**; a value *must* be specified.

### 4.6.7 align

**Align** specifies the horizontal alignment of text in a table cell. The **align** attribute has the following predefined values: "left", "right", "center", "justify", and "char" (Section 4.6.8). The default value is "left" for **TGroup** and "center" for **SpanSpec**. The value for all others is typically inherited from **TGroup**.

### 4.6.8 char

**Char** specifies the character on which to align when an element has an **align** value of "char." For example, if a table group contained rows of numbers that had decimal-point alignment, then the following markup would be used:

```
<TGroup align="char" char=".">...</TGroup>
```

Values for **char** can be inherited from parent elements.

### 4.6.9 charoff

**Charoff** specifies where in the horizontal space of the column the alignment character (**char**, Section 4.6.8) should be. The value of **charoff** is a number that represents a percentage of the column width. The default value is "50" (centered).

### 4.6.10 valign

**valign** specifies the vertical alignment of text within a table row. There are three predefined values for **valign**: "top", "middle", and "bottom". The default **valign** value is "top" for **TBody** and **TFoot**, and "bottom" for **THead**.

### 4.6.11 tabstyle

**Tabstyle** is used to indicate the name of an externally defined style (presumably in a FOSI) from which the table can inherit formatting defaults. There are no predefined values.

### 4.6.12 tgroupstyle

**Tgroupstyle** is used to indicate the name of an externally defined style (presumably in a FOSI) from which the table group can inherit formatting defaults. There are no predefined values.

## 4.7 Preferred order of attributes

To simplify the task of converting TIM files into other markup, a preferred order of attributes has been established. The preferred order is followed in the ATTLIST declarations of the TIM DTD, but the overall order for all attributes in all elements is provided here for reference. Document originators are encouraged to write the attributes needed within element start-tags in the following order:

```
 1 type
 2 id
 3 dstype
 4 remap
 5 remapatt
 6 remapto
 7 group
 8 status
 9 version
10 revstat
11 lang
12 present
13 presspec
14 emph
15 prefer
16 dimensions
17 placement
18 application
19 format
20 sortas
21 label
22 keywords
23 alerts
24 altreps
25 close
26 entityref
27 linkends
28 spanend
29 rid
30 url
31 sortorder
32 elements
```

```
33 levels
34 types
35 remaps
36 groups
37 statuses
38 revstats
39 langs
40 scope
41 target
42 numeration
43 prefix
44 infix
45 suffix
46 include
47 inheritnum
48 inheritfrom
49 splitnum
50 continuation
51 restartsat
52 increment
53 shownum
54 symbolseq
55 segments
56 segwidths
57 separator
58 lastsep
59 arrange
60 columns
61 width
62 TIM2
63 HyTime
64 HyNames
65 anchrole
66 extra
67 intra
68 refrange
69 locsrc
70 quantum
71 catsrc
72 catres
73 nametype
74 obnames
75 docorsub
76 dtdorlpd
```

# 5 Using TIM

## 5.1 Which TIM?

There are two files that are both considered the TIM DTD. Still others may be derived from those. Which should you use?

- *tim2.dtd* is the reference version, the "official" DTD. *tim2.dtd* has the public identifier:

  ```
  -//USA/TCIF//DTD TIM-2//EN
  ```

  That identifier should appear in the prologue of any TIM document instance that needs extensions to TIM, but ...

- *tim2.hdr* is the more readable version, but it's not, strictly speaking, a DTD. It has an SGML declaration at the beginning. If you remove that, it's exactly equivalent to *tim2.dtd*. Aside from being the version for humans to look at, its main purpose is for quick validation with some of the free parsers that are available (specifically *sgmls*, see Section 5.4). You have to comment out the first line of your document instance:

  ```
  <!--DOCTYPE TDoc "-//USA/TCIF//DTD TIM-2//EN" [-->
  ```

  Then you can concatenate *tim2.hdr* with your instance to make a complete, parsable SGML document that requires no external files. For instance, in MS-DOS you can say:

  ```
  C:\SGMLS>sgmls -s -ferror.txt tim2.hdr mydoc.tim
  ```

  And you'll get a list of all parsing errors (in the file *error.txt*).

Any other versions of TIM are not standard. But they may be useful. As an interchange DTD, TIM allows for almost any document structure that any telecom company uses in its technical documents. That's too much for any one company to allow in its own documents. The main goal of any customized version of TIM is to restrict the allowable structures to something like what is actually done in a given company's documents, to simplify authoring with SGML-aware applications or conversion from other applications. Since the restrictions will be company-specific, so will the DTDs. Any such DTD that a company is willing to share may be found in a *contrib* subdirectory of some of the TCIF **ftp** and **http** sites listed in Section 5.4. Use what's there only as a starting point for your own implementation.

## 5.2 Updates and version numbering

Updates to the two versions of the DTD are always released simultaneously. The IPI Committee follows this policy for TIM updates and version numbering:

- A major revision is one that will invalidate most existing TIM document instances (that is, the revised TIM DTD will contain rules that earlier documents violate, so that an SGML parser will find errors where there were none before). Any such major revision must be submitted to the TCIF

membership for approval by letter ballot. If a major revision is approved, the first part of the version number will be incremented (that is, the next major revision will be called Release 3.0.0). No such revision shall be submitted for approval until there are exact specifications for programs that update older document instances to make them valid, and at least one instance of such a program has been written and tested successfully. (Such a program exists for the conversion of TIM1 instances to TIM2.) Every effort will be made to limit major releases to no more than one every two years.

- A minor revision is one that meaningfully enhances TIM, such as by adding new elements or attributes, without invalidating most existing document instances. Minor revisions that are deemed by a consensus of the IPI Committee to be likely to invalidate only a small fraction of existing TIM document instances shall require approval only by a consensus of the IPI Committee. If there is no consensus within the IPI Committee that only a few TIM instances will be invalidated, the revision must be submitted to the TCIF membership for approval by letter ballot. If a minor revision is approved by either process, the second part of the version number will be incremented (that is, the next minor revision will be called Release 2.1.0). No such revision shall be submitted for approval until a good-faith effort has been made to identify document instances that will be invalidated and provide instructions for updating them. Every effort will be made to limit minor releases to no more than one every six months.
- A maintenance revision is one that corrects an error or omission in TIM that keeps it from performing according to the intentions stated in this Guideline. These revisions may be made at any time either by the IPI Technical Team Leader or by a consensus of the IPI Technical Team, subject to review by the IPI Committee as a whole. A maintenance revision must be deemed to be unlikely to invalidate any existing TIM document instances that follow the intent of these Guidelines. A maintenance revision is indicated by incrementing the third part of the version number (that is, the next maintenance revision will be called Release 2.0.1). No such revision shall be released until a good-faith effort has been made to determine whether any document instances would be invalidated.
- If there is no consensus within the IPI Committee on which level of revision is appropriate, the procedure for the higher level will be followed.

It is important for both producers and recipients of TIM documents to make sure that they are using the most recent release of the TIM DTD. Recipients can expect that at least some of the documents they receive will include the latest markup, which an older release of the DTD might not allow parsers and applications to recognize. Producers should also stay up-to-date, because even if a revision is considered minor or maintenance — assumed to invalidate few or no existing document instances — that judgment is based on examination of instances existing at the time. A producer who creates new instances with an older version of the DTD may be creating instances that are invalid in new and unforeseen ways.

Conversely, there is little danger in using the latest DTD. Except after major revisions, which will be rare and well-publicized, older document instances will normally work with the newest version of the DTD. We fully expect that most Release 2.0.0 document instances will be parsable with Release 2.9.99 of the TIM DTD, if there ever is one.

The way to stay up-to-date is to download the latest TIM files from one of the sites listed in Section 5.4. To be put on the e-mailing list to be informed of each update, send e-mail to:

```
majordomo@listserv.atis.org
```

with this as the body of the message:

```
subscribe ipi-list
```

## 5.3 Getting started with TIM

### 5.3.1 Viewing TIM documents

Because TIM documents conform to the SGML standard, they can be viewed with any SGML-compliant browser (such as SoftQuad's Panorama), or any browser that can convert SGML input to its own format (there are many of those). There are four things you will usually need to provide to the browser in order to view a TIM document:

- The file *tcif.dcl*. This is the SGML declaration; it provides the browser with the information it needs to process the TIM DTD.
- The file *tim2.dtd*. This is the document type declaration; it provides the browser with the information it needs to interpret the structure of the TIM document. You also need *calstbl2.dtd*, *tcif_kb.ent* and the various ISO entity files. You can find them all at the same locations.
- A stylesheet. This is a file that provides the browser with the information it needs to display the document in a familiar way. It specifies fonts, point sizes, colors, margins, indents, and the other things that make the document look right. You will probably want a stylesheet that produces a familiar appearance, and you may have to create this yourself.
- At least one TEDD package containing a TIM document instance plus any supplemental files (such as graphics).

Generally, the declaration, DTD, and stylesheet files need only be installed for the browser one time. Once they are installed, the browser will be able to read in and format any valid TIM 2 document instance. Stylesheets for various browsers may be available at the TCIF URL addresses (Section 5.4), along with the other necessary files.

#### 5.3.1.1 Recipients' responsibilities

The producers of TIM documents may expect that the recipient's presentation tools meet certain technical requirements, to ensure that the presentation is

complete and accurate. These requirements could reasonably include the following:

- All text content of all elements in a TIM file must be available to the user, except (optionally) text within:
  - **`Contents`**, **`Index`**, **`Ref`**, **`URLRef`**, and **`IndexEntry`** elements
  - elements having a **`revstat`** attribute with the value "delete" or "strike"
  - any elements within a marked section.
- All available text content must be presented (i.e., visible in a visual presentation or audible in an aural presentation) by default, except (optionally) text within elements with the **`present`** attribute set to "float" or "altrep".
- All nontext content provided in valid files using formats approved by TCIF (in TCIF-IPI-96-004 and future guidelines) must be available to the user by some means.
- All contents of elements with the present attribute set to "alert" must be presented in a way that attracts the user's attention (with distinctive sound or color, graphics or borders, heavy or enlarged type, etc.).
- All text content marked for emphasis (with the **`Emph`** element or attribute) must be presented distinctively (e.g., italic, bold, underlined, large, distinctly colored, blinking, or marked by special text or symbols).

#### 5.3.1.2 Recipients' prerogatives

Document producers should recognize that stylistic aspects of presentation will be under the control of the recipient. These include:

- Size of page or presentation window
- Type sizes, faces, and fonts, and type and background colors
- Column widths, indents, and vertical spacing
- Numbering and labeling of elements (sections, list items, etc.)
- Content and style of cross-references, indexes, and lists of contents

and, generally,

- Interpretation of attribute values with respect to presentation.

### 5.3.2 Producing TIM documents

Preparing to produce TIM documents is likely to require more effort and expense than preparing to use them. A TIM document is a plain ASCII file with special tags inserted in it to denote document structure, special characters, external entities, etc. Because a TIM file itself contains nothing but ASCII text, no special software is required to create one. A simple TIM document may actually be as easy to create with a basic text editor like **vi**, TeachText, or Notepad as with a

specialized SGML authoring tool. To apply TIM markup to an existing document, one could conceivably scan a paper copy with OCR software and add appropriate tags either manually or automatically according to recognizable formatting features.

However, a real technical document needs a lot of markup, and much of it must be exactly right. For any more than a few small documents, the practical approach to production involves writing TIM-specific conversion filters, a task for programmers. The conversion will be easier or more complicated, and will require more or less manual cleanup (or maybe none at all), depending on what markup your documents are created in.

#### 5.3.2.1 Production environments

Although it is possible to do so, the IPI Committee does not recommend creating documents directly with TIM markup. And in any case, documents that already exist were created with some other markup. To set up a production environment to deliver TIM documents regularly, the alternatives, from worst to best, are:

1. Allow document originators to use any word-processing or desktop-publishing system they like, and set up a conversion department that, mostly manually, applies TIM markup after-the-fact. This would increase costs and delay delivery beyond what almost any company could tolerate.
2. Still allow document originators to use any word-processing or desktop-publishing system they like, but arrange with a document-conversion service provider to create the TIM files. This would at least make the costs and delivery delays smaller and more predictable. Conversion specialists are probably the most practical choice for large-scale conversion of legacy documents, if the documents are fairly homogeneous (mostly created with the same software and the same templates or stylesheets). The specialists can quickly write filters to convert all the markup that's used consistently.
3. Standardize on any one word-processing or desktop-publishing application for deliverable documentation, supply everyone with the same templates or stylesheets, and strictly enforce consistent use of them. In such an environment, conversion can be mostly automated, but there will always be some manual cleanup.
4. Allow document originators to use any *SGML* authoring application they like (a "native" SGML application like Author Editor or ArborText, a hybrid like FrameMaker+SGML, or an add-on to a regular application like SGML Author for MS Word or WordPerfect SGML), and supply them all with the TIM DTD and the application-specific stylesheets and templates that each would need. In this environment, writers would be creating TIM files directly, so there would be no conversion cost or lag time. But because of the nature of TIM, there would also be very little consistency in the documentation (even if everyone used the same SGML application). TIM, as an interchange DTD, allows authors to do too many things, in too many different ways, to let a company produce consistent documentation.

5. Create you own authoring DTD, consistent with both TIM and your company's publications standards (your documents do or don't have a preface; the main divisions are Chapters or Sections; subsections are numbered or not; you do or don't allow a list with just one item; etc.). Standardize on an SGML authoring application *or* allow document originators to use any SGML authoring application they like (this doesn't matter very much, since everyone is still creating the same kind of file; the trade-off is how many different systems need to be supported vs. how many value-added features in those systems can be exploited).

This order of preference is affected very little by the factors that used to influence decisions about a document-production environment: size of the operation, budget constraints, hardware standards already in place, or other required outputs. For instance, if you need to produce Acrobat PDF or HTML output as well as TIM and paper, these can be created easily, if not from the applications themselves then by conversion from your SGML or from TIM — conversion from structure-oriented markup like TIM to layout-oriented markup like HTML and PDF is virtually 100% automatable, as is conversion between two structure-oriented DTDs (conversion *into* structure-oriented markup is not so easy).

### 5.3.2.2 Creating an authoring DTD

Any SGML authoring system requires a DTD, and we recommend that you use a DTD more appropriate for authoring than TIM. If you don't already have an authoring DTD, we suggest that you become familiar with TIM and then, starting there, create YOUR DTD this way:

- Tighten up content models to reflect what you actually allow in your documents. For instance, the TIM content model for **`Section`** allows an optional title and then any number of paragraph-level elements, nested **`Section`**s, and **`Contents`** and **`Index`** lists, in any order:

  ```
  <!ELEMENT (Body|FrontMatter|AppMatter|BackMatter|Section) -
  - (TitleGroup?,(Section|Contents|Index|P|S|...)*) >
  ```

  You might prefer that a **`Section`** always have a title, that a **`Contents`** list within a **`Section`** occur only at the beginning, that there be no **`Index`** lists (except as part of back matter), that once there is a nested **`Section`** there can be no more para-level elements that aren't in a nested **`Section`**, and that there must be at least two nested **`Section`**s if there are any. So your DTD might have this model:

  ```
  <!ELEMENT Section - - (TitleGroup, Contents?, (P|S|...)*,
  (Section, Section+)?) >
  ```

You might have rules even more specific for **`FrontMatter`**, and so on. You could even remove some elements from every content model if you'll never use them. Note that, with only these kinds of changes, YOUR DTD is 100% compatible with TIM: any document instance that is valid according to YOUR DTD is also valid

according to the much less restrictive TIM DTD. (Of course, someone else's TIM instance might not be valid with YOUR DTD.)

- Limit the allowable values of attributes (especially **type** and the attributes for numbering) to those that are meaningful in your documents. (Most SGML-aware authoring application can format elements according to attribute values.) For instance, if the only types of **Annote**s in your documents are footnotes, "Important" and "Reminder" notes, and "Author's Notes," you could require that each **Annote** be one of those:

  ```
  <!ATTLIST Annote     type
  (FNote|Important|Reminder|AuthorNote) #REQUIRED
  ```

  You may remove some attributes completely because you won't use them. YOUR DTD will still be 100% compatible with TIM.

- If your application cannot format according to attribute values, or if your authors find it too cumbersome to select the right attributes, you can turn any element+attribute combination into a distinct element. For instance, if your ordered lists can be only numbered or upper-case-lettered, you might want to convert **<OrderedList numeration=arabic ...>** to **<NumList>** and **<OrderedList numeration=upperalpha ...>** to **<AlphaList>**. YOUR DTD will no longer be 100% compatible with the TIM DTD, but the conversion will be very, very simple.

- If TIM's various wrapper elements are confusing to your application or your authors, you can remove them. For instance, if your authors have to insert a **TitleGroup** and *then* insert a **Title** every time they need to type a title (some applications will automatically insert required children), just remove **TitleGroup** from YOUR DTD. You may decide you can do without list groups, and you may find that there are too many levels of elements in TIM's **Table**s and **SegmentedList**s for your application's table formatter to handle. The conversion to and from TIM will still be straightforward, because the places to put wrapper start and end tags are never ambiguous. (**TitleGroup**, in fact, exists mainly to make conversions easier, by reducing the number of possible element sequences a filter can encounter.)

- If you will be creating a document database to maximize the usefulness and reusability of your information assets, you may want to add specific elements for types of information that your internal document users will want to search for or reuse often. You might, for instance, copy the content model of **DistOrdLI** into a new element you call **Requirement**, which is to be used always and only for system requirements. Then a query of your database could be restricted to **Requirement**s (you might be able to do the same thing with attribute values, so new elements might not be necessary). As long as you do not invent new content models for the new elements, the conversion to and from TIM will be just like the conversion of list types described above, but in the opposite direction.

## 5.4 Where to get things: SGML software, TIM files and documentation, updates and announcements, and other TCIF Guidelines on electronic documentation

All material produced by the TCIF IPI Committee concerning TIM is available at one or more of these sites:

- **http://www.atis.org/atis/tcif/ipi**
- **ftp://ftp.bellcore.com/pub/world/TCIF** (command-line ftp only)
- **ftp://ftp.isogen.com/pub/tcif** (accessible with Web browsers)

There are several subdirectories at the first three sites. TIM files are in *tim*. Documentation, including TIM, text, and PostScript versions of this document, are in *ipi_tedd/ipi95004*.

The following free and low-cost software is readily available at other Internet sites. (Generally, software is not provided at the TCIF sites because it is hard to make sure everything is the latest version.)

- **sgmls** (freeware SGML parser)

  `ftp://ftp.ifi.uio.no/pub/SGML/SGMLS`

  A parser is used to check the validity of a document's structure against the rules of the DTD — a TIM document must be validated by a parser before it can be used by an SGML rendering application. Most SGML software already has a parser built into it; however, if you do not have any SGML software, **sgmls** can be used as a stand-alone parser to check your TIM documents.

- Panorama (browser from SoftQuad, Inc.)

  `http://www.sq.com/products/panorama/panor-fe.htm`

  A browser for SGML websites that also allows you to publish SGML documents as websites or information archives. Both commercial and free versions are available. The free version can be used only with a World Wide Web browser, such as Mosaic. Only the MS Windows version is available as of this writing, but Macintosh and UNIX versions have been promised.

- MS Windows graphics viewers to use with Panorama

  `ftp://oak.oakland.edu/simtel/win3/graphics`

Descriptions of some of the graphics viewers available are in the *ipi_tool/graphics* subdirectory at some of the TCIF Internet sites. Most are shareware, with registration fees between $15 and $69.

- **perl** (freeware text-manipulation utility)

  for UNIX:

  `ftp://ftp.uu.net/gnu`
  `ftp://archive.cis.ohio-state.edu/perl`

```
ftp://jpl-devvax.jpl.nasa.gov/pub
```

for MS-DOS:

```
ftp://ftp.ee.umanitoba.ca/pub/msdos/perl
ftp://ftp.cis.ufl.edu/pub/perl/src/msdos
ftp://ftp.funet.fi/pub/languages/perl/ports/perl4/msdos
ftp://src.doc.ic.ac.uk/packages/ibmpc/simtel/perl/
```

This a good choice for developing a conversion program to go from word-processing or non-TIM SGML markup to TIM without using commercial software. Commercial packages like Exoterica's OmniMark provide routines and interfaces that simplify the writing of filters to and from SGML. But, generally speaking, programming experience is valuable, and training necessary, if you're going to write filters with any of the available tools. Some filters are likely to be made available at the TCIF sites; look in the *ipi_tool* directory. Each filter will be script for a particular application like **perl** or OmniMark; you'll need the application to be able to use the filter.

## 5.5 Where to send things: questions, contributions, and comments

Questions about TIM or SGML or these guidelines should be sent to:

```
dfp@ims.bellcore.com
```

You may also call 908-699-4012, but we prefer e-mail since we're building a Question-and-Answer archive from the messages we receive. ASCII files (text, TIM, or **uuencode**d or mime-encoded binaries) may also be e-mailed to the addresses above. Files that you'd rather **ftp** should be uploaded to:

```
ftp://ftp.bellcore.com/incoming
```

Please send e-mail or call to let us know when you've uploaded something. When necessary, the FAX number is 908-336-3711, and the mail address is:

Don Pratt

Bellcore

8 Corporate Place, PYA-3K140

Piscataway, NJ 08854

Suggestions or comments about TIM or these Guidelines, and questions about the other work of the TCIF IPI Committee should be directed to its chair, Art Kedzierski, 817-540-8296, `artkedz@airmail.net` (vice-chair: Tonua Brown, 615-734-4728, `tonua_brown@nt.com`).

## 5.6 SGML bibliography

The references and resources listed below cover SGML in general. References specifically on TIM are listed in Section 5.4. Software is also listed there.

### 5.6.1 Paper-based references

1. DeRose, Steven J., and Durand, David G., *Making Hypermedia Work, A User's Guide to HyTime*. Boston: Kluwer Academic Publishers, 1994.

   A guide to the HyTime standard (ISO 10744). This book explains the concepts of hypertext and hypermedia, and how HyTime can be used to incorporate them into documents.

2. Goldfarb, Charles, *The SGML Handbook*. New York: Oxford University Press, 1990.

   A fully annotated version of the SGML standard (ISO 8879). It is intended to be an explication of SGML syntax for SGML application developers; it is *not* intended for end users and does not provide any information about how to actually use an SGML application or write SGML documents.

3. *The SGML Guide*. Waltham: Interleaf, Inc., 1994.

   Provides a high-level overview of the concepts of SGML and its various benefits. This booklet is intended for people who must make a business case for the adoption of SGML within their company.

4. Travis, B., and Waldt, D., *The SGML Implementation Guide, A Blueprint for SGML Migration*. Berlin: Springer-Verlag, 1995.

   A complete guide to SGML implementation; from document analysis to setting up an SGML application.

5. van Herwijnen, Eric, *Practical SGML, Second Edition*. Boston: Kluwer Academic Publishers, 1994.

   A detailed SGML reference. This book is designed to help novices become proficient in SGML. It can also serve as a quick reference to experienced SGML users.

### 5.6.2 CD-ROM references

1. *SGML World Tour*. SoftQuad, Inc., 1994.

   A compilation of articles, FAQs, DTDs, and other SGML resources by members of the SGML OPEN consortium. Runs on Windows 3.x, DOS, Macintosh, or UNIX (Sun 4.1.x or Solaris 1).

2. *SGML Power Tools*. The SGML University, 1995.

   A collection of SGML tools, utilities, tutorials, and research materials. Requires Windows 3.x, Windows NT, or Windows '95.

### 5.6.3 On-line resources

1. *The SGML Primer*. Toronto: SoftQuad, Inc. 1995.

   `http://www.sq.com/sgmlinfo/primintr.html`

2. The SGML Web Page

   `http://www.sil.org/sgml/sgml.html`

A starting point for exploring the body of SGML-related information available on the internet. This page has embedded links to WWW, ftp, and gophers servers that provide information about SGML.

3. SGML on the Web

   `http://www.ncsa.uiuc.edu/WebSGML/WebSGML.html`

   A guide to sites and services that provide SGML documents over the World Wide Web.

4. SGML OPEN consortium

   `http://www.sgmlopen.org/`

   A nonprofit, international consortium of suppliers whose products and services support SGML.

5. The SGML University

   `http://www.sgml.com/`

   A virtual university that provides sources of information, events, directories, job listings, and other information critical to the SGML implementor and student.

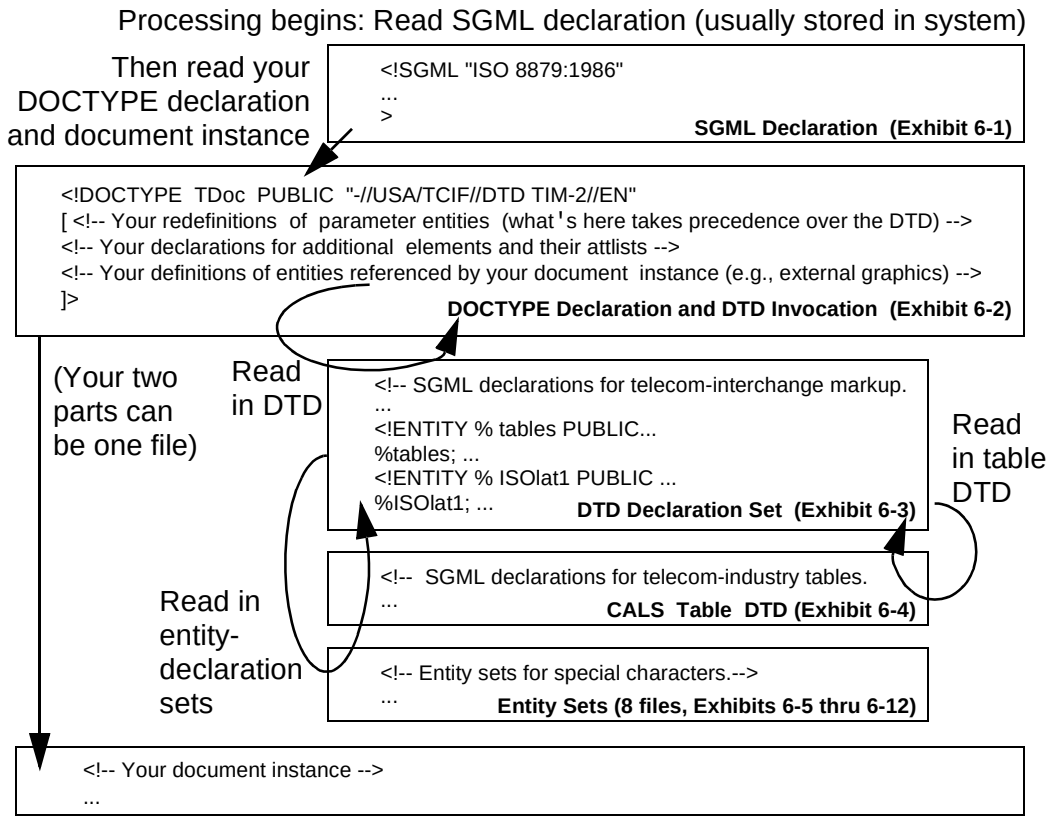
# 6 The TIM Document Type Definition

This section provides listings of all SGML declarations needed to create and validate TIM documents. As shown in Figure 6-1, the SGML declarations for the reference version of TIM are packaged in several separate files:

- one containing the SGML declaration used for TIM markup
- one containing a doctype declaration and DTD invocation
- one containing the core declarations, referred to as the TIM DTD
- a separate one containing the declarations defining tables
- eight containing entities for non-ASCII characters

(All of these files are combined in *tim2.hdr*.) The modularization of the declaration sets provides more flexibility for making local modifications to the DTD. The doctype declaration calls in the TIM DTD declaration set, and it calls in the others through public identifiers. Each SGML application must keep track of public identifiers and map them to particular files.

**Figure 6-1** A TIM Document



The entity references for special characters other than keycaps characters are in ISO-standard files provided with virtually any SGML application, but they are reproduced in Exhibits Exhibit 6-5 through Exhibit 6-12 for the sake of completeness. Sites from which all the files may be downloaded are listed in Section 5.4.

## 6.1 The SGML Declaration

The TIM Document Type Definition uses an SGML declaration recommended by the Information Products Interchange (IPI) Committee of the Telecommunications Industry Forum (TCIF), in related document IPI-95-001. The TCIF declaration upgrades many SGML quantities and capacities beyond those listed in ISO 8879 for a "basic SGML document." Exhibit 6-1 contains the TCIF SGML declaration.

**Exhibit 6-1** The TCIF SGML Declaration (line numbers are not part of the file)

```
<<!SGML "ISO 8879:1986"
 -- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    TCIF SGML Declaration.
    Version: 2.0.0, 1997-02-05 (tcif2.dcl)
    This declaration must appear first in the data stream constituting a
    conforming TCIF SGML document, but it can be assumed in a conforming
    XML document. An SGML data stream should also include a TCIF DTD.
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ --
CHARSET   BASESET "ISO 646-1983//CHARSET International Reference Version
                  (IRV)//ESC 2/5 4/0"
DESCSET   0   9  UNUSED
          9   2  9
         11   2  UNUSED
         13   1  13
         14  18  UNUSED
         32  95  32
        127   1  UNUSED
--BASESET "ISO Registration Number 109//CHARSET ECMA-94 Right Part of Latin
         Alphabet Nr. 3 //ESC 2/13 4/3"
DESCSET 128  32  UNUSED
        160   5  32
        165  89  32
        254   1  127
        255   1  UNUSED             for XML--
 -- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    Characters 0-31 and 127 are control characters, and the DESCSET
    section specifies that only tab (9), linefeed (10), and carriage
    return (13) are meaningful. ("0   9  UNUSED" means 9 characters
    beginning with #0 are not used meaningfully in the data stream.)
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ --
CAPACITY  SGMLREF
    TOTALCAP  99999999
    ENTCAP    99999999
    ENTCHCAP  99999999
    ELEMCAP   99999999
    GRPCAP    99999999
    EXGRPCAP  99999999
    EXNMCAP   99999999
    ATTCAP    99999999
    ATTCHCAP  99999999
    AVGRPCAP  99999999
```

```
    NOTCAP    99999999
    NOTCHCAP  99999999
    IDCAP     99999999
    IDREFCAP  99999999
    MAPCAP    99999999
    LKSETCAP  99999999
    LKNMCAP   99999999
 -- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    Capacities have been increased from the "reference capacity set", so
    that large documents won't automatically generate parsing errors.
    (In "basic SGML documents," capacities are set at 35,000.)
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ --
SCOPE     DOCUMENT
 -- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    The scope of this declaration is the entire document. Any DTD or other
    prolog you include must conform to it, as well as document content -
    not hard, since this declaration is less restrictive than the default.
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ --
SYNTAX
 -- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    This declaration does not use the "reference concrete syntax," so it
    can't necessarily be processed by a "conforming SGML application."
    You may need a system that is specifically XML-conforming.
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ --
SHUNCHAR  NONE
BASESET   "ISO 646-1983//CHARSET International Reference Version
           (IRV)//ESC 2/5 4/0"
DESCSET   0    128   0
FUNCTION  RE        13
          RS        10
          SPACE     32
          TAB       SEPCHAR  9
NAMING    LCNMSTRT  ""       -- "-" for XML--
          UCNMSTRT  ""       -- "-" for XML--
          LCNMCHAR  "-."     -- "." for XML--
          UCNMCHAR  "-."     -- "." for XML--
          NAMECASE  GENERAL YES
                    ENTITY   NO
DELIM     GENERAL   SGMLREF
             -- NET   "/>"          for XML--
             -- PIC   "?>"          for XML--
          SHORTREF  SGMLREF
NAMES     SGMLREF
QUANTITY  SGMLREF
          NAMELEN         64     -- for XML, all these 99999999--
          LITLEN        1024
          ATTCNT         128
          GRPCNT         128
          GRPGTCNT       253
          TAGLVL         128
          ATTSPLEN      2000
          ENTLVL          64
          GRPLVL          16
          PILEN         2048
          TAGLEN        2048
 -- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    Names (of elements, attributes, etc.) may be 64 characters, not just 8.
    There may be 128 attributes in an ATTLIST, not just 40. There may be
    253 (not 32) tokens in a group, and 253 (not 96) in a content model.
    Elements may be nested to 128 levels, not 24. A literal (a parameter
    or attribute value in quotes) may be 1024 characters, not just 240.
    Names (of elements, attributes, etc.) are automatically converted to
    upper case, except for entities.
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ --
```

```
FEATURES
MINIMIZE  DATATAG  NO  OMITTAG   NO  RANK      NO    SHORTTAG  YES
LINK      SIMPLE   NO  IMPLICIT  NO  EXPLICIT  NO
OTHER     CONCUR   NO  SUBDOC    NO  FORMAL    NO
 -- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    OMITTAG and SHORTTAG are used in "basic SGML documents."
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ --
APPINFO   NONE
 -- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    No application-specific information is specified.
 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->
```

## 6.2 The DOCTYPE Declaration and DTD Invocation

The most common doctype used in TIM markup will be the **TDoc** doctype. The TIM DTD declaration set itself, however, contains no doctype declaration so that the declarations may be used with other doctype declarations when desired. A doctype declaration, along with specifications of entity reference sets used in document instances, can be placed in a file separate from the file containing the TIM DTD declarations and can invoke the TIM DTD declarations through the use of a public or system identifier. Alternatively, although less conveniently, a doctype declaration may be placed directly in the file containing the TIM DTD declarations. Probably the most convenient packaging is to make this invocation a prologue of each TIM document instance. Exhibit 6-2 provides a typical TIM DTD invocation file that uses the TelDoc doctype.

**Exhibit 6-2** Typical invocation file (doctype declaration)

```
<!DOCTYPE TDoc PUBLIC "-//USA/TCIF//DTD TIM-2//EN" [
-- SGML DOCTYPE declaration and TIM DTD invocation.
     Release: @(#) 2.0.0 97/07/31 (tim2.ivk) --
-- Put your entity declarations (redefinitions of the entities in the
   TIM DTD and declarations for external entities such as graphics files)
   here. Everything inside the square brackets is read before the
   TIM DTD and takes precedence over it --
-- Put your declarations for additional elements here. This file should
   be preceded by the TCIF SGML declaration and followed by your document
   instance. All other files needed for processing are invoked directly
   or indirectly by this file. --
]>
```

## 6.3 The TIM DTD Declaration Set

Exhibit 6-3 lists the TIM DTD (formally, the TIM DTD declaration set). TCIF-IPI-95-004-PART2 describes the elements and attributes in alphabetical order.

**Exhibit 6-3** The TIM DTD Declaration Set

```
<!--*SGML declarations for Telecommunications Interchange Markup/Technical
     Interchange Markup (TIM), release 2.B.12, 1997-07-17 (file: tim2.dtd)*-->
<!--*Copyright (c) 1995-1997 Alliance for Telecommunications Industry
     Solutions ("ATIS"). Early versions copyright (c) 1994-1995 Bellcore and
     ATIS. Title to the copyright for its contribution is transferred by
     Bellcore jointly to Bellcore and ATIS, with the understanding that they
     jointly own the copyright in such contribution without either party's
```

```
     accounting to the other therefor.*-->
<!--*Portions of this DTD have been derived from the DocBook DTD, Revision
     2.1, copyright (c) 1992, 1993 HaL Computer Systems International, Ltd.,
     and O'Reilly & Associates, Inc.*-->
<!--*You may use, copy, modify, and distribute this material for any purpose,
     without fee, provided that this notice appears in all copies and that the
     names of Bellcore and ATIS do not appear in advertising or publicity
     pertaining to this material without the specific, prior written permission
     of authorized representatives of ATIS and Bellcore. Reproduction and
     distribution for resale is prohibited.*-->
<!--*NO REPRESENTATIONS ARE MADE ABOUT THE SUITABILITY OF THIS MATERIAL FOR ANY
     PURPOSE. IT IS PROVIDED "AS IS" WITHOUT EXPRESS OR IMPLIED WARRANTIES.*-->
<!--*Applicability: This is the TCIF (Telecommunications Industry Forum)
     SGML/XML DTD for interchange of technical documents within the telecom
     industry. It is not necessarily appropriate for authoring applications,
     archival storage of information, or direct input to browsers. It most
     likely needs extensions to efficiently mark up documents with more-
     formalized structures than those in general telecom practices, such as
     procedures or requirements, or for exchange of information structured in
     other ways than as discrete documents.*-->
<!--*Updates, related files, and documentation will be available at:
       http://www.atis.org/atis/tcif/ipi
     Additional sites for downloading:
       ftp://ftp.bellcore.com/pub/world/TCIF (command-line ftp only)
       ftp://ftp.isogen.com/pub/tcif (accessible with Web browsers)
     TIM 2 is defined and described in TCIF-IPI-95-004-Part1, Issue 2, and
     TCIF-IPI-95-004-Part2, Issue 2. To order, contact TCIF, 202-434-8844.*-->
<!--*SECTION 1. NOTATIONS*-->
<!NOTATION Animator  SYSTEM "Animator" >
<!NOTATION AU        SYSTEM "AU" >
<!NOTATION AVI       SYSTEM "AVI" >
<!NOTATION BMP       PUBLIC "+//ISBN 0-7923-9432-1::Graphic Notation//NOTATION
                             Microsoft Windows bitmap//EN" >
<!NOTATION CGM       SYSTEM "CGM-BINARY" >
<!NOTATION CGM-CLEAR PUBLIC "ISO 8632/4//NOTATION Clear text encoding//EN" >
<!NOTATION DOC       SYSTEM "Microsoft Word document">
<!NOTATION DXF       SYSTEM "DXF" >
<!NOTATION EPS       PUBLIC "+//ISBN 0-201-18127-4::Adobe//NOTATION
                             PostScript Language Ref. Manual//EN" >
<!NOTATION EQN       SYSTEM "EQN" >
<!NOTATION FileBasename PUBLIC "-//TCIF-IPI-95-004::FileBasename//NOTATION
                             File basename//EN" >
<!NOTATION GIF       PUBLIC "+//ISBN 0-7923-9432-1::Graphic Notation//NOTATION
                             CompuServe Graphic Interchange Format//EN" >
<!NOTATION HTML      SYSTEM "HTML" >
<!NOTATION JFIF      PUBLIC "+//ISBN 0-7923-9432-1::Graphic Notation//NOTATION
                             JPEG File Interchange Format//EN" >
<!NOTATION MIDI      SYSTEM "MIDI" >
<!NOTATION MOV       SYSTEM "MOV" >
<!NOTATION MPEG      SYSTEM "MPEG" >
<!NOTATION PBM       PUBLIC "+//ISBN 0-7923-9432-1::Graphic Notation//NOTATION
                             Jef Poskanzer Portable Bit Map//EN" >
<!NOTATION PDF       SYSTEM "PDF" >
<!NOTATION PGM       SYSTEM "PGM" >
<!NOTATION PIC       SYSTEM "PIC" >
<!NOTATION PICT      PUBLIC "+//ISBN 0-7923-9432-1::Graphic Notation//NOTATION
                             Apple Computer QuickDraw Picture//EN" >
<!NOTATION PNG       SYSTEM "PNG" >
<!NOTATION PPM       SYSTEM "PPM" >
<!NOTATION PPT       SYSTEM "PPT presentation">
<!NOTATION PS        SYSTEM "PS" >
<!NOTATION QUICKTIME SYSTEM "QUICKTIME" >
<!NOTATION RTF       SYSTEM "RTF" >
<!NOTATION SGML      PUBLIC "ISO 8879:1986//NOTATION Standard Generalized
```

```
                            Markup Language//EN" >
<!NOTATION SUNRASTER PUBLIC "+//ISBN 0-7923-9432-1::Graphic Notation//NOTATION
                            Sun Microsystems raster//EN" >
<!NOTATION TBL       SYSTEM "TBL" >
<!NOTATION TEX       PUBLIC "+//ISBN 0-201-13448-9::Knuth//NOTATION
                            The TeXbook//EN" >
<!NOTATION TEXT      SYSTEM "Plain text" >
<!NOTATION TIFF      PUBLIC "+//ISBN 0-7923-9432-1::Graphic Notation//NOTATION
                            Aldus/Microsoft
                            Tagged Interchange File Format//EN" >
<!NOTATION TIFFGRP4  SYSTEM "Black & white TIFF with Group IV compression" >
<!NOTATION WAV       SYSTEM "WAV" >
<!NOTATION WK1       SYSTEM "WK1 spreadsheet" >
<!NOTATION WMF       PUBLIC "+//ISBN 0-7923-9432-1::Graphic Notation//NOTATION
                            Microsoft Windows Metafile//EN" >
<!NOTATION WPG       SYSTEM "WPG" >
<!NOTATION XLS       SYSTEM "XLS spreadsheet">
<!NOTATION XWD       PUBLIC "+//ISBN 0-7923-9432-1::Graphic Notation//NOTATION
                            MIT X Consortium Window Dump//EN" >
<!--*SECTION 2. ENTITIES FOR ATTRIBUTE GROUPS AND FOR POSSIBLE EXTENSIONS*-->
<!ENTITY % baseatts
         "type NMTOKENS #IMPLIED
          dstype CDATA #IMPLIED
          remap NMTOKEN #IMPLIED
          remapatt CDATA #IMPLIED
          remapto CDATA #IMPLIED
          group NMTOKENS #IMPLIED
          status CDATA #IMPLIED
          version IDREFS #IMPLIED
          revstat (norev|revised|revflag|insert|insflag|delete|strike) #IMPLIED
          lang CDATA #IMPLIED
          present (normal|float|alert|altrep) #IMPLIED
          presspec CDATA #IMPLIED
          TIM2 NMTOKEN #IMPLIED">
<!ENTITY % identity
         "id ID #IMPLIED
          label CDATA #IMPLIED
          keywords CDATA #IMPLIED">
<!ENTITY % numbering
         "numeration (nonum|arabic|outline|upperalpha|loweralpha|upperroman
           |lowerroman|symbol) #IMPLIED
          prefix CDATA #IMPLIED
          infix CDATA #IMPLIED
          suffix CDATA #IMPLIED
          inheritnum (inherit|inherit0|inherit1|inherit2|inherit3|noinherit)
           #IMPLIED
          inheritfrom NMTOKENS #IMPLIED
          splitnum (nosplit|splitarabic|splitualpha|splitlalpha|spliturоman
           |splitlroman|splitoutline) #IMPLIED
          continuation (continues|restarts|holds) #IMPLIED
          restartsat NMTOKEN #IMPLIED
          increment NMTOKEN #IMPLIED
          shownum NMTOKEN #IMPLIED
          symbolseq CDATA #IMPLIED">
<!ENTITY % commonatts
         "emph (ital|bold|boldital|roman|und|dblund|color|sys) #IMPLIED
          alerts IDREFS #IMPLIED
          altreps IDREFS #IMPLIED">
<!ENTITY % textatts
         "format NOTATION (SGML|HTML|RTF|TEX|TBL|EQN|PIC|EPS|PS|PBM|PGM|PPM
           |CGM-CLEAR|TEXT) #IMPLIED
          sortas CDATA #IMPLIED">
<!ENTITY % tables PUBLIC "-//USA/TCIF//ELEMENTS Modified CALS Tables 2//EN">
<!ENTITY % eqns         "">
```

```
<!ENTITY % flows        "">
<!ENTITY % procs        "">
<!ENTITY % rqmts        "">
<!ENTITY % memos        "">
<!ENTITY % other        "">
%other;
%memos;
%rqmts;
%procs;
%eqns;
%flows;
%tables;
<!--*SECTION 3. ELEMENTS AND THEIR ATTRIBUTES*-->
<!--*3.A. HIERARCHICAL STRUCTURES*-->
<!ELEMENT TDoc - - (DocID,Resources?,FrontMatter?,Body,AppMatter?,BackMatter?)>
<!ATTLIST TDoc
          %baseatts;
          %identity;>
<!ELEMENT (Body|FrontMatter|AppMatter|BackMatter|Section) - - ((Title
          |TitleGroup)?,(Contents|Index|P|S|Annote|Frame|Danger|Caution
          |Warning|Admon|DistOrdLI|DistVarLI|Table|OrderedList|UnorderedList
          |VariableList|SegmentedList|Listing|Pt|IndexTerm|Graphic|Object
          |ExternalText|Flowchart|Section)*)>
<!ATTLIST (Body|FrontMatter|AppMatter|BackMatter)
          %baseatts;
          %identity;
          alerts IDREFS #IMPLIED
          altreps IDREFS #IMPLIED>
<!ATTLIST Section
          %baseatts;
          %identity;
          %numbering;
          %commonatts;>
<!ELEMENT Resources - - (Link|NameLoc|DataLoc|TreeLoc)*>
<!ATTLIST Resources
          %baseatts;
          %identity;>
<!ELEMENT Contents - - ((Title|TitleGroup)?,(P|S|Annote|Frame|Danger|Caution
          |Warning|Admon|Table|OrderedList|UnorderedList|VariableList
          |SegmentedList|Listing|Pt|IndexEntry|Graphic|Object|ExternalText
          |Flowchart|Contents)*)>
<!ATTLIST Contents
          %baseatts;
          %identity;
          elements NMTOKENS #IMPLIED
          levels NMTOKENS #IMPLIED
          groups NMTOKENS #IMPLIED
          types NMTOKENS #IMPLIED
          remaps CDATA #IMPLIED
          langs CDATA #IMPLIED
          revstats NMTOKENS #IMPLIED
          statuses CDATA #IMPLIED
          versions CDATA #IMPLIED
          scope NMTOKENS #IMPLIED
          include NMTOKENS #IMPLIED>
<!ELEMENT Index - - ((Title|TitleGroup)?,(P|S|Annote|Frame|Danger|Caution
          |Warning|Admon|Table|OrderedList|UnorderedList|VariableList
          |SegmentedList|Listing|Pt|IndexEntry|Graphic|Object|ExternalText
          |Flowchart|Index)*)>
<!ATTLIST Index
          %baseatts;
          %identity;
          sortorder CDATA #IMPLIED
          elements NMTOKENS #IMPLIED
```

```
200           levels NMTOKENS #IMPLIED
201           groups NMTOKENS #IMPLIED
202           types NMTOKENS #IMPLIED
203           remaps CDATA #IMPLIED
204           langs CDATA #IMPLIED
205           revstats NMTOKENS #IMPLIED
206           statuses CDATA #IMPLIED
207           versions CDATA #IMPLIED
208           scope NMTOKENS #IMPLIED
209           include NMTOKENS #IMPLIED>
210 <!--*3.B. PARAS AND BLOCK LISTS*-->
211 <!ELEMENT P - - (#PCDATA|T|Sub|Sup|Emph|Symbol|Ref|URLRef|SimpleList
212           |IndexTerm|Pt|Graphic|Object|ExternalText|Equation|GraphicalText
213           |S|Annote|Frame|Danger|Caution|Warning|Admon|Table|OrderedList
214           |UnorderedList|VariableList|SegmentedList|Flowchart|Listing)*>
215 <!ATTLIST P
216           %baseatts;
217           %identity;
218           %numbering;
219           %commonatts;
220           %textatts;>
221 <!ELEMENT (ListTerm|SimpleLI) - - (#PCDATA|T|Sub|Sup|Emph|Symbol|Ref
222           |URLRef|Pt|IndexTerm|Graphic|Object|ExternalText|Equation
223           |GraphicalText)*>
224 <!ATTLIST (ListTerm|SimpleLI)
225           %baseatts;
226           %identity;
227           %commonatts;
228           %textatts;>
229 <!ELEMENT LI - - (Title?,(P|S|Annote|Frame|Danger|Caution|Warning|Admon
230           |Table|OrderedList|UnorderedList|VariableList|SegmentedList
231           |Listing|Pt|IndexTerm|Graphic|Object|ExternalText|Flowchart)*)>
232 <!ELEMENT VarLI - - (ListTerm+,Pt*,LI,(Pt|LI)*)>
233 <!ELEMENT SegLI - - (Pt*,LI,(Pt|LI)*)>
234 <!ATTLIST (LI|VarLI|SegLI)
235           %baseatts;
236           %identity;
237           %commonatts;>
238 <!ELEMENT SimpleList - - (SimpleLI+)>
239 <!ATTLIST SimpleList
240           %baseatts;
241           %identity;
242           %numbering;
243           %commonatts;
244           separator CDATA #IMPLIED
245           lastsep CDATA #IMPLIED
246           arrange (inline|vert|horiz) #IMPLIED
247           columns NMTOKEN #IMPLIED>
248 <!ELEMENT SegmentedList - - ((Title|TitleGroup)?,(SegListGrp+))>
249 <!ATTLIST SegmentedList
250           %baseatts;
251           %identity;
252           %commonatts;
253           colsep NMTOKEN #IMPLIED
254           frame (top|bottom|topbot|all|sides|none) #IMPLIED
255           rowsep NMTOKEN #IMPLIED
256           liststyle NMTOKEN #IMPLIED
257           segments NMTOKEN #REQUIRED
258           segwidths CDATA #IMPLIED>
259 <!ELEMENT SegListGrp - - (ListHead?,(Annote|Danger|Caution|Warning|Admon|SegLI
260           |Pt)*)>
261 <!ELEMENT UnorderedList - - ((Title|TitleGroup)?,ListHead?,(UnordListGrp+
262           |(Danger|Caution|Warning|Admon|Annote|LI|Pt)*))>
263 <!ELEMENT UnordListGrp - - (Title?,ListHead?,(Danger|Caution|Warning|Admon
```

```
264           |Annote|LI|Pt)*)>
265 <!ATTLIST (UnorderedList|UnordListGrp)
266           %baseatts;
267           %identity;
268           mark CDATA #IMPLIED
269           %commonatts;>
270 <!ELEMENT OrderedList - - ((Title|TitleGroup)?,ListHead?,(OrdListGrp+|(Danger
271           |Caution|Warning|Admon|Annote|LI|Pt)+))>
272 <!ELEMENT OrdListGrp - - (Title?,ListHead?,(Danger|Caution|Warning|Admon
273           |Annote|LI|Pt)*)>
274 <!ELEMENT VariableList - - ((Title|TitleGroup)?,ListHead?,(VarListGrp+|(Danger
275           |Caution|Warning|Admon|Annote|VarLI|Pt)*))>
276 <!ELEMENT VarListGrp - - (Title?,ListHead?,(Danger|Caution|Warning|Admon
277           |Annote|VarLI|Pt)*)>
278 <!ATTLIST (SegListGrp|OrderedList|OrdListGrp|VariableList|VarListGrp)
279           %baseatts;
280           %identity;
281           %numbering;
282           %commonatts;>
283 <!ELEMENT ListHead - - (Title+)>
284 <!ATTLIST ListHead
285           %baseatts;>
286 <!ELEMENT Listing  - - (#PCDATA|T|Sub|Sup|Emph|Symbol|Ref|URLRef
287           |IndexTerm|Pt|Graphic|Object|ExternalText)*>
288 <!ATTLIST Listing
289           %baseatts;
290           %identity;
291           %numbering;
292           %commonatts;
293           %textatts;
294           width NUMBER #IMPLIED>
295 <!--*3.C. NON-HIERARCHICAL STRUCTURES*-->
296 <!ELEMENT S - - ((Title|TitleGroup)?,(P|S|Annote|Frame|Danger|Caution|Warning
297           |Admon|DistOrdLI|DistVarLI|Table|OrderedList|UnorderedList
298           |VariableList|SegmentedList|Listing|Pt|IndexTerm|Graphic|Object
299           |ExternalText|Flowchart)*)>
300 <!ATTLIST S
301           %baseatts;
302           %identity;
303           %numbering;
304           %commonatts;>
305 <!ELEMENT DistOrdLI - - (Title?,ListHead?,(P|S|Annote|Frame|Danger|Caution
306           |Warning|Admon|Table|OrderedList|UnorderedList|VariableList
307           |SegmentedList|Listing|Pt|IndexTerm|Graphic|Object|ExternalText
308           |Flowchart)*)>
309 <!ELEMENT DistVarLI - - (Title?,ListHead?,ListTerm+,Pt*,LI,(Pt|LI)*)>
310 <!ATTLIST (DistOrdLI|DistVarLI)
311           %baseatts;
312           %identity;
313           %numbering;
314           %commonatts;>
315 <!ELEMENT Frame - - ((Title|TitleGroup)?,(P|S|Annote|Frame|Danger|Caution
316           |Warning|Admon|DistOrdLI|DistVarLI|Table|OrderedList
317           |UnorderedList|VariableList|SegmentedList|Listing|Pt
318           |IndexTerm|Graphic|Object|ExternalText|Equation
319           |GraphicalText|Flowchart)*)>
320 <!ATTLIST Frame
321           %baseatts;
322           %identity;
323           %numbering;
324           %commonatts;
325           dimensions NMTOKENS #IMPLIED
326           placement NMTOKENS #IMPLIED
327           valign (top|middle|bottom) #IMPLIED
```

```
328           align (left|right|center) #IMPLIED
329           layer NMTOKEN #IMPLIED>
330 <!ELEMENT (Danger|Caution|Warning|Admon) - - (Title?,(P|S|Annote|Frame|Table
331           |OrderedList|UnorderedList|VariableList|SegmentedList|Listing|Pt
332           |IndexTerm|Graphic|Object|ExternalText|Flowchart)*)>
333 <!ATTLIST (Danger|Caution|Warning|Admon)
334           %baseatts;
335           %identity;
336           %numbering;
337           %commonatts;>
338 <!ELEMENT Annote - - (Title?,(P|S|Annote|Frame|Danger|Caution|Warning|Admon
339           |Table|OrderedList|UnorderedList|VariableList|SegmentedList
340           |Listing|Pt|IndexTerm|Graphic|Object|ExternalText|Flowchart)*)>
341 <!ATTLIST Annote
342           %baseatts;
343           %identity;
344           %numbering;
345           %commonatts;>
346 <!--*3.D. TITLES*-->
347 <!ELEMENT TitleGroup - - (SuperTitle*,Title+,SubTitle*,ShortTitle*)>
348 <!ATTLIST TitleGroup
349           %baseatts;>
350 <!ELEMENT (Title|SuperTitle|SubTitle|ShortTitle) - - (#PCDATA|T|Sub|Sup|Emph
351           |Symbol|Ref|URLRef|IndexTerm|Pt|Graphic|Object|ExternalText
352           |SimpleList|Equation|GraphicalText)*>
353 <!ATTLIST (Title|SuperTitle|SubTitle|ShortTitle)
354           %baseatts;
355           %commonatts;
356           %textatts;>
357 <!--*3.E. PHRASE-LEVEL ELEMENTS*-->
358 <!ELEMENT T - - (#PCDATA|T|Sub|Sup|Emph|Symbol|Ref|URLRef|IndexTerm|Pt
359           |Graphic|Object|ExternalText|SimpleList|Equation|GraphicalText)*>
360 <!ELEMENT (Equation|GraphicalText) - - (#PCDATA|T|Sub|Sup|Emph|Symbol|Ref
361           |URLRef|IndexTerm|Pt|Graphic|Object|ExternalText|SimpleList|Equation
362           |GraphicalText)*>
363 <!ATTLIST (T|Equation|GraphicalText)
364           %baseatts;
365           %identity;
366           %commonatts;
367           %textatts;>
368 <!ELEMENT (Sub|Sup|Emph) - - (#PCDATA|T|Sub|Sup|Emph|Symbol|Ref|URLRef
369           |IndexTerm|Pt|Graphic|Object|ExternalText|SimpleList|Equation
370           |GraphicalText)*>
371 <!ELEMENT Symbol - - (#PCDATA)>
372 <!ATTLIST (Sub|Sup|Emph|Symbol)
373           %baseatts;
374           id ID #IMPLIED
375           emph (ital|bold|boldital|roman|und|dblund|color|sys)
376             #IMPLIED
377           %textatts;>
378 <!--*3.F. LINKS, LOCATIONS, AND INDEX TERMS*-->
379 <!ELEMENT Link - O  EMPTY>
380 <!ATTLIST Link
381           %baseatts;
382           %identity;
383           remap.anchrole NMTOKENS #IMPLIED
384           remap.extra NMTOKENS #IMPLIED
385           remap.intra NMTOKENS #IMPLIED
386           remap.refrange CDATA #IMPLIED
387           linkends IDREFS #REQUIRED
388           HyTime NMTOKEN #FIXED "ilink"
389           anchrole NMTOKENS #FIXED "from to"
390           extra NMTOKENS "A A"
391           intra NMTOKENS "A A"
```

```
392           refrange CDATA #FIXED "#ALL X">
393 <!ELEMENT Pt - O EMPTY>
394 <!ATTLIST Pt
395           %baseatts;
396           %identity;
397           %numbering;
398           %commonatts;
399           close NMTOKEN #IMPLIED
400           spanend IDREF #IMPLIED
401           HyTime NMTOKEN #FIXED "clink"
402           HyNames CDATA #FIXED "linkend spanend"
403           refrange NMTOKENS "spanend X">
404 <!ELEMENT NameLoc - - (NmList+)>
405 <!ATTLIST NameLoc
406           id ID #REQUIRED
407           HyTime NMTOKEN #FIXED "NameLoc">
408 <!ELEMENT NmList - - (#PCDATA)>
409 <!ATTLIST NmList
410           HyTime NMTOKEN #FIXED "NmList"
411           nametype (entity|element|unified) "element"
412           obnames (obnames|nobnames) "nobnames"
413           docorsub ENTITY #IMPLIED
414           dtdorlpd NMTOKENS #IMPLIED>
415 <!ELEMENT TreeLoc - - (#PCDATA)>
416 <!ATTLIST TreeLoc
417           id ID #REQUIRED
418           HyTime NMTOKEN #FIXED "TreeLoc"
419           locsrc IDREFS #IMPLIED>
420 <!ELEMENT DataLoc - - (#PCDATA)>
421 <!ATTLIST DataLoc
422           id ID #REQUIRED
423           HyTime NMTOKEN #FIXED "DataLoc"
424           locsrc IDREFS #IMPLIED
425           quantum (str|norm|word|name|sint|date|time|utc) "norm"
426           catsrc (catsrc|catsrcsp|nocatsrc) "nocatsrc"
427           catres (catres|catressp|nocatres) "nocatres">
428 <!ELEMENT (Ref|URLRef) - - (#PCDATA|T|Sub|Sup|Emph|Symbol|Ref|URLRef|IndexTerm
429           |Pt|Graphic|Object|ExternalText|SimpleList|Equation|GraphicalText)*>
430 <!ATTLIST Ref
431           %baseatts;
432           %identity;
433           %commonatts;
434           %textatts;
435           rid IDREF #REQUIRED
436           target NMTOKEN #IMPLIED
437           prefix CDATA #IMPLIED
438           infix CDATA #IMPLIED
439           suffix CDATA #IMPLIED
440           include NMTOKENS #IMPLIED
441           HyTime NMTOKEN #FIXED "clink"
442           HyNames CDATA #FIXED "linkend rid"
443           refrange NMTOKENS "rid X">
444 <!ATTLIST URLRef
445           %baseatts;
446           %identity;
447           %commonatts;
448           %textatts;
449           url CDATA #REQUIRED
450           target NMTOKEN #IMPLIED
451           prefix CDATA #IMPLIED
452           infix CDATA #IMPLIED
453           suffix CDATA #IMPLIED
454           include NMTOKENS #IMPLIED>
455 <!ELEMENT (IndexTerm|IndexEntry) - - (#PCDATA|T|Sub|Sup|Emph|Symbol|Ref
```

```
456           |URLRef|Equation|GraphicalText|IndexEntry|Pt)*>
457 <!ATTLIST (IndexTerm|IndexEntry)
458           %baseatts;
459           %commonatts;
460           %textatts;
461           id ID #IMPLIED
462           prefer (preferred|ordinary) #IMPLIED>
463 <!ELEMENT (ExternalText|Graphic|Object|Flowchart) - O EMPTY>
464 <!ATTLIST ExternalText
465           %baseatts;
466           %identity;
467           %commonatts;
468           entityref ENTITY #REQUIRED>
469 <!ATTLIST (Graphic|Flowchart)
470           %baseatts;
471           %identity;
472           %commonatts;
473           dimensions NMTOKENS #IMPLIED
474           placement NMTOKENS #IMPLIED
475           valign (top|middle|bottom) #IMPLIED
476           align (left|right|center) #IMPLIED
477           layer NMTOKEN #IMPLIED
478           entityref ENTITY #REQUIRED>
479 <!ATTLIST Object
480           %baseatts;
481           %identity;
482           %commonatts;
483           dimensions NMTOKENS #IMPLIED
484           placement NMTOKENS #IMPLIED
485           valign (top|middle|bottom) #IMPLIED
486           align (left|right|center) #IMPLIED
487           layer NMTOKEN #IMPLIED
488           application CDATA #IMPLIED
489           entityref ENTITY #REQUIRED>
490 <!--*3.G. DOCUMENT IDENTIFICATION*-->
491 <!ELEMENT DocID - - (EdocID,Class,Company,(DocNo|DocNo.u),DocDate,TitleGroup,
492           VersionControl?,AltNo*,Publisher?,ISN*,Country*,Copyrt?,Lang*,
493           (SuperDoc|SubDoc)*,CDocType?,MTextType,DraftStatus?,Status*,
494           PropStat?,PropMsg?,EdocRepl*,DocRepl*,ProdDsc*,DocDsc?,RelDoc*,
495           OrderInfo?,Contact*,Au*,Abs?,Keywords?)>
496 <!ELEMENT Company - - (#PCDATA|Graphic)*>
497 <!ELEMENT DocNo - - (DN.base,(DN.iss|DN.ed|DN.addm|DN.rev|DN.rvl|DN.vol|DN.apdx
498           |DN.bull|DN.supp|DN.rlse|DN.lang|DN.date|DN.prod|DN.country
499           |DN.customer)*)>
500 <!ELEMENT VersionControl - - (Version)+>
501 <!ELEMENT Version - - (Au*,VersDate,VersDesc*)>
502 <!ELEMENT (EdocID|Class|DocNo.u|DocDate|DN.base|DN.iss|DN.ed|DN.addm|DN.rev
503           |DN.rvl|DN.vol| DN.apdx|DN.bull|DN.supp|DN.rlse|DN.lang|DN.date
504           |DN.prod|DN.country|DN.customer|AltNo|Publisher|ISN|Status|Copyrt
505           |PropStat|PropMsg|Lang|Country|ProdDsc|DocDsc|EdocRepl|OrderInfo|Au
506           |Contact|Abs|Keywords|DraftStatus|CDocType|MTextType|VersDate
507           |VersDesc) - - (#PCDATA)*>
508 <!ELEMENT (SuperDoc|SubDoc|RelDoc|DocRepl) - - ((DocNo|DocNo.u),
509           (Title|TitleGroup)?)>
510 <!ATTLIST (DocID|EdocID|Class|Company|DocNo|DocNo.u|DN.base|DN.iss|DN.ed
511           |DN.addm|DN.rev|DN.rvl|DN.vol|DN.apdx|DN.bull|DN.supp|DN.rlse|DN.lang
512           |DN.date|DN.prod|DN.country|DN.customer|DocDate|AltNo|DraftStatus
513           |Publisher|ISN|Country|Copyrt|Lang|SuperDoc|CDocType|SubDoc|Status
514           |PropStat|PropMsg|EdocRepl|DocRepl|ProdDsc|DocDsc|RelDoc|OrderInfo
515           |Contact|Au|Abs|Keywords|MTextType|VersionControl|VersDate|VersDesc)
516           id ID #IMPLIED
517           remap NMTOKEN #IMPLIED
518           remapatt CDATA #IMPLIED
519           remapto CDATA #IMPLIED
```

```
520           TIM2 NMTOKEN #IMPLIED>
521 <!ATTLIST Version
522           id ID #REQUIRED
523           use (include|ignore) #IMPLIED
524           remap NMTOKEN #IMPLIED
525           remapatt CDATA #IMPLIED
526           remapto CDATA #IMPLIED
527           TIM2 NMTOKEN #IMPLIED>
528 <!--*SECTION 4. ENTITIES FOR SPECIAL CHARACTERS*-->
529 <!ENTITY % ISOlat1 PUBLIC "ISO 8879-1986//ENTITIES Added Latin 1//EN">
530 <!ENTITY % ISOlat2 PUBLIC "ISO 8879-1986//ENTITIES Added Latin 2//EN">
531 <!ENTITY % ISOgrk3 PUBLIC "ISO 8879-1986//ENTITIES Greek Symbols//EN">
532 <!ENTITY % ISOnum PUBLIC "ISO 8879-1986//ENTITIES
533            Numeric and Special Graphic//EN">
534 <!ENTITY % ISObox PUBLIC "ISO 8879-1986//ENTITIES Box and Line Drawing//EN">
535 <!ENTITY % ISOtech PUBLIC "ISO 8879-1986//ENTITIES General Technical//EN">
536 <!ENTITY % ISOpub PUBLIC "ISO 8879-1986//ENTITIES Publishing//EN">
537 <!ENTITY % TCIFkb PUBLIC "-//USA/TCIF//ENTITIES Keybd-2//EN">
538 %ISOlat1;
539 %ISOlat2;
540 %ISOgrk3;
541 %ISOnum;
542 %ISObox;
543 %ISOtech;
544 %ISOpub;
545 %TCIFkb;
546 <!--*SECTION 5. PROCESSING INSTRUCTIONS*-->
547 <?HyTime VERSION "ISO/IEC 10744:1992" HYQCNT=32>
548 <?HyTime MODULE base>
549 <?HyTime MODULE locs>
550 <?HyTime MODULE links>
551 <?ATTLINK URLRef url URI>
552 <?XML ECODING='UTF-8'>
```

## 6.4 The CALS Table DTD

Exhibit 6-4 shows the TCIF version of the CALS Table DTD, which is the default table model for TIM markup. It is invoked by the **`%tables;`** entity reference in line 147 of the TIM DTD.

This DTD would normally be stored as a separate file, and the entity reference above would be interpreted by the processing system to call in this file. Alternatively, this could be included directly in an SGML document data stream by replacing line 147 of the TIM DTD with this file.

**Exhibit 6-4** The CALS Table DTD

```
 1 <!--*SGML declarations for TIM tables.
 2      Version 2.0.4, 1997-07-17 (calstbl2.dtd)*-->
 3 <!--*This is the TCIF version of the CALS table DTD, derived from MIL-M-28001B
 4      via DocBook. Entry has been given a content comparable to that of TIM
 5      Frame. There are no security, ShortEntry, or Tocentry attributes.*-->
 6 <!ENTITY % tblatts
 7           "type NMTOKENS #IMPLIED
 8            dstype CDATA #IMPLIED
 9            remap NMTOKEN #IMPLIED
10            remapatt CDATA #IMPLIED
11            remapto CDATA #IMPLIED
12            group NMTOKENS #IMPLIED
13            status CDATA #IMPLIED
```

```
          version IDREFS #IMPLIED
          revstat (norev|revised|revflag|insert|insflag|delete|strike) #IMPLIED
          lang CDATA #IMPLIED
          present (normal|float|alert|altrep) #IMPLIED
          presspec CDATA #IMPLIED
          TIM2 NMTOKEN #IMPLIED">
<!ELEMENT Table - - ((Title|TitleGroup)?,TGroup+)>
<!ATTLIST Table
          id ID #IMPLIED
          %tblatts;
          label CDATA #IMPLIED
          keywords CDATA #IMPLIED
          numeration (nonum|arabic|outline|upperalpha|loweralpha
            |upperroman|lowerroman|symbol) #IMPLIED
          prefix CDATA #IMPLIED
          infix CDATA #IMPLIED
          suffix CDATA #IMPLIED
          inheritnum (inherit|inherit0|inherit1|inherit2
            |inherit3|noinherit) #IMPLIED
          inheritfrom NMTOKENS #IMPLIED
          splitnum (nosplit|splitarabic|splitualpha|splitlalpha
            |spliturоman|splitlroman|splitoutline) #IMPLIED
          continuation (continues|restarts|holds) #IMPLIED
          restartsat NMTOKEN #IMPLIED
          increment NMTOKEN #IMPLIED
          shownum NMTOKEN #IMPLIED
          symbolseq CDATA #IMPLIED
          emph (ital|bold|boldital|roman|und|dblund|color|sys) #IMPLIED
          alerts IDREFS #IMPLIED
          altreps IDREFS #IMPLIED
          colsep NMTOKEN #IMPLIED
          frame (top|bottom|topbot|all|sides|none) #IMPLIED
          orient (port|land) #IMPLIED
          pgwide NMTOKEN #IMPLIED
          rowsep NMTOKEN #IMPLIED
          tabstyle NMTOKEN #IMPLIED>
<!ELEMENT TGroup - - (ColSpec*,SpanSpec*,THead?,TFoot?,TBody)>
<!ATTLIST TGroup
          id ID #IMPLIED
          %tblatts;
          align (left|right|center|justify|char) "left"
          char CDATA ""
          charoff NMTOKEN "50"
          cols NMTOKEN #REQUIRED
          colsep NMTOKEN #IMPLIED
          rowsep NMTOKEN #IMPLIED
          tgroupstyle NMTOKEN #IMPLIED>
<!ELEMENT (ColSpec|SpanSpec) - O EMPTY>
<!ATTLIST ColSpec
          id ID #IMPLIED
          remap NMTOKEN #IMPLIED
          remapatt CDATA #IMPLIED
          remapto CDATA #IMPLIED
          align (left|right|center|justify|char) #IMPLIED
          char CDATA #IMPLIED
          charoff NMTOKEN #IMPLIED
          colname NMTOKEN #IMPLIED
          colnum NMTOKEN #IMPLIED
          colsep NMTOKEN #IMPLIED
          colwidth CDATA #IMPLIED
          rowsep NMTOKEN #IMPLIED
          TIM2 NMTOKEN #IMPLIED>
<!ATTLIST SpanSpec
          id ID #IMPLIED
```

```
          remap NMTOKEN #IMPLIED
          remapatt CDATA #IMPLIED
          remapto CDATA #IMPLIED
          align  (left|right|center|justify|char) "center"
          char CDATA #IMPLIED
          charoff NMTOKEN #IMPLIED
          colsep NMTOKEN #IMPLIED
          nameend NMTOKEN #REQUIRED
          namest NMTOKEN #REQUIRED
          rowsep NMTOKEN #IMPLIED
          spanname NMTOKEN #REQUIRED
          TIM2 NMTOKEN #IMPLIED>
<!ELEMENT (THead|TFoot) - - (ColSpec*,Pt*,Row,(Pt|Row)*)>
<!ATTLIST THead
          id ID #IMPLIED
          %tblatts;
          valign (top|middle|bottom) "bottom">
<!ELEMENT TBody - - (Pt*,Row,(Pt|Row)*)>
<!ATTLIST (TBody|TFoot)
          id ID #IMPLIED
          %tblatts;
          valign (top|middle|bottom) "top">
<!ELEMENT Row - - (Pt*,(Entry|EntryTbl),(Pt|Entry|EntryTbl)*)>
<!ATTLIST Row
          id ID #IMPLIED
          %tblatts;
          rowsep NMTOKEN #IMPLIED
          valign (top|middle|bottom) #IMPLIED>
<!ELEMENT Entry - - (P|S|Annote|Frame|Danger|Caution|Warning|Admon|DistOrdLI
          |DistVarLI|OrderedList|UnorderedList|VariableList|Listing|Pt
          |IndexTerm|Graphic|Object|Flowchart|ExternalText|Equation
          |GraphicalText)*>
<!ATTLIST Entry
          id ID #IMPLIED
          %tblatts;
          align (left|right|center|justify|char) #IMPLIED
          char CDATA #IMPLIED
          charoff NMTOKEN #IMPLIED
          colname NMTOKEN #IMPLIED
          colsep NMTOKEN #IMPLIED
          morerows NMTOKEN "0"
          nameend NMTOKEN #IMPLIED
          namest NMTOKEN #IMPLIED
          rotate NMTOKEN "0"
          rowsep NMTOKEN #IMPLIED
          spanname NMTOKEN #IMPLIED
          valign (top|middle|bottom) #IMPLIED>
<!ELEMENT EntryTbl - - (ColSpec*,SpanSpec*,THead?,TBody)>
<!ATTLIST EntryTbl
          id ID #IMPLIED
          %tblatts;
          align (left|right|center|justify|char) #IMPLIED
          char CDATA #IMPLIED
          charoff NMTOKEN #IMPLIED
          colname NMTOKEN #IMPLIED
          cols NMTOKEN #REQUIRED
          colsep NMTOKEN #IMPLIED
          nameend NMTOKEN #IMPLIED
          namest NMTOKEN #IMPLIED
          rowsep NMTOKEN #IMPLIED
          spanname NMTOKEN #IMPLIED
          tgroupstyle NMTOKEN #IMPLIED>
```

## 6.5 Entity sets for special characters

The TIM document type declaration (Exhibit 6-3) calls in several entity-reference sets, all but one of them ISO-standard sets that virtually all SGML applications already include. The one that may have to be installed for your application is the TCIF entity set for Keycaps. These entity references can be used to represent the keyboard keys other than the numbers, letters, and punctuation marks: s, e, 1, and so on. The alternative is to use regular characters in a **<Symbol>** element, which signals to the processing application that each character or entity within the element should be rendered in a special font.

**Exhibit 6-5** Entity set for box-drawing characters (iso-box.ent)

```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISObox PUBLIC
       "ISO 8879-1986//ENTITIES Box and Line Drawing//EN">
     %ISObox;
-->
<!-- All names are in the form: box1234, where:
    box = constants that identify a box drawing entity.
    1&2 = v, V, u, U, d, D, Ud, or uD, as follows:
      v = vertical line for full height.
      u = upper half of vertical line.
      d = downward (lower) half of vertical line.
    3&4 = h, H, l, L, r, R, Lr, or lR, as follows:
      h = horizontal line for full width.
      l = left half of horizontal line.
      r = right half of horizontal line.
    In all cases, an upper-case letter means a double or heavy line.
-->
<!ENTITY boxh   SDATA "[boxh  ]"--horizontal line -->
<!ENTITY boxv   SDATA "[boxv  ]"--vertical line-->
<!ENTITY boxur  SDATA "[boxur ]"--upper right quadrant-->
<!ENTITY boxul  SDATA "[boxul ]"--upper left quadrant-->
<!ENTITY boxdl  SDATA "[boxdl ]"--lower left quadrant-->
<!ENTITY boxdr  SDATA "[boxdr ]"--lower right quadrant-->
<!ENTITY boxvr  SDATA "[boxvr ]"--upper and lower right quadrants-->
<!ENTITY boxhu  SDATA "[boxhu ]"--upper left and right quadrants-->
<!ENTITY boxvl  SDATA "[boxvl ]"--upper and lower left quadrants-->
<!ENTITY boxhd  SDATA "[boxhd ]"--lower left and right quadrants-->
<!ENTITY boxvh  SDATA "[boxvh ]"--all four quadrants-->
<!ENTITY boxvR  SDATA "[boxvR ]"--upper and lower right quadrants-->
<!ENTITY boxhU  SDATA "[boxhU ]"--upper left and right quadrants-->
<!ENTITY boxvL  SDATA "[boxvL ]"--upper and lower left quadrants-->
<!ENTITY boxhD  SDATA "[boxhD ]"--lower left and right quadrants-->
<!ENTITY boxvH  SDATA "[boxvH ]"--all four quadrants-->
<!ENTITY boxH   SDATA "[boxH  ]"--horizontal line-->
<!ENTITY boxV   SDATA "[boxV  ]"--vertical line-->
<!ENTITY boxUR  SDATA "[boxUR ]"--upper right quadrant-->
<!ENTITY boxUL  SDATA "[boxUL ]"--upper left quadrant-->
<!ENTITY boxDL  SDATA "[boxDL ]"--lower left quadrant-->
<!ENTITY boxDR  SDATA "[boxDR ]"--lower right quadrant-->
<!ENTITY boxVR  SDATA "[boxVR ]"--upper and lower right quadrants-->
<!ENTITY boxHU  SDATA "[boxHU ]"--upper left and right quadrants-->
<!ENTITY boxVL  SDATA "[boxVL ]"--upper and lower left quadrants-->
```

```
<!ENTITY boxHD  SDATA "[boxHD ]"--lower left and right quadrants-->
<!ENTITY boxVH  SDATA "[boxVH ]"--all four quadrants-->
<!ENTITY boxVr  SDATA "[boxVr ]"--upper and lower right quadrants-->
<!ENTITY boxHu  SDATA "[boxHu ]"--upper left and right quadrants-->
<!ENTITY boxVl  SDATA "[boxVl ]"--upper and lower left quadrants-->
<!ENTITY boxHd  SDATA "[boxHd ]"--lower left and right quadrants-->
<!ENTITY boxVh  SDATA "[boxVh ]"--all four quadrants-->
<!ENTITY boxuR  SDATA "[boxuR ]"--upper right quadrant-->
<!ENTITY boxUl  SDATA "[boxUl ]"--upper left quadrant-->
<!ENTITY boxdL  SDATA "[boxdL ]"--lower left quadrant-->
<!ENTITY boxDr  SDATA "[boxDr ]"--lower right quadrant-->
<!ENTITY boxUr  SDATA "[boxUr ]"--upper right quadrant-->
<!ENTITY boxuL  SDATA "[boxuL ]"--upper left quadrant-->
<!ENTITY boxDl  SDATA "[boxDl ]"--lower left quadrant-->
<!ENTITY boxdR  SDATA "[boxdR ]"--lower right quadrant-->
```

**Exhibit 6-6** Entity set for Greek symbols (iso-grk3.ent)

```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOgrk3 PUBLIC
       "ISO 8879-1986//ENTITIES Greek Symbols//EN">
     %ISOgrk3;
-->
<!ENTITY alpha    SDATA "[alpha ]"--=small alpha, Greek-->
<!ENTITY beta     SDATA "[beta  ]"--=small beta, Greek-->
<!ENTITY gamma    SDATA "[gamma ]"--=small gamma, Greek-->
<!ENTITY Gamma    SDATA "[Gamma ]"--=capital Gamma, Greek-->
<!ENTITY gammad   SDATA "[gammad]"--/digamma-->
<!ENTITY delta    SDATA "[delta ]"--=small delta, Greek-->
<!ENTITY Delta    SDATA "[Delta ]"--=capital Delta, Greek-->
<!ENTITY epsi     SDATA "[epsi  ]"--=small epsilon, Greek-->
<!ENTITY epsiv    SDATA "[epsiv ]"--/varepsilon-->
<!ENTITY epsis    SDATA "[epsis ]"--/straightepsilon-->
<!ENTITY zeta     SDATA "[zeta  ]"--=small zeta, Greek-->
<!ENTITY eta      SDATA "[eta   ]"--=small eta, Greek-->
<!ENTITY thetas   SDATA "[thetas]"--straight theta-->
<!ENTITY Theta    SDATA "[Theta ]"--=capital Theta, Greek-->
<!ENTITY thetav   SDATA "[thetav]"--/vartheta - curly or open theta-->
<!ENTITY iota     SDATA "[iota  ]"--=small iota, Greek-->
<!ENTITY kappa    SDATA "[kappa ]"--=small kappa, Greek-->
<!ENTITY kappav   SDATA "[kappav]"--/varkappa-->
<!ENTITY lambda   SDATA "[lambda]"--=small lambda, Greek-->
<!ENTITY Lambda   SDATA "[Lambda]"--=capital Lambda, Greek-->
<!ENTITY mu       SDATA "[mu    ]"--=small mu, Greek-->
<!ENTITY nu       SDATA "[nu    ]"--=small nu, Greek-->
<!ENTITY xi       SDATA "[xi    ]"--=small xi, Greek-->
<!ENTITY Xi       SDATA "[Xi    ]"--=capital Xi, Greek-->
<!ENTITY pi       SDATA "[pi    ]"--=small pi, Greek-->
<!ENTITY piv      SDATA "[piv   ]"--/varpi-->
<!ENTITY Pi       SDATA "[Pi    ]"--=capital Pi, Greek-->
<!ENTITY rho      SDATA "[rho   ]"--=small rho, Greek-->
<!ENTITY rhov     SDATA "[rhov  ]"--/varrho-->
<!ENTITY sigma    SDATA "[sigma ]"--=small sigma, Greek-->
<!ENTITY Sigma    SDATA "[Sigma ]"--=capital Sigma, Greek-->
<!ENTITY sigmav   SDATA "[sigmav]"--/varsigma-->
<!ENTITY tau      SDATA "[tau   ]"--=small tau, Greek-->
<!ENTITY upsi     SDATA "[upsi  ]"--=small upsilon, Greek-->
```

```
<!ENTITY Upsi     SDATA "[Upsi  ]"--=capital Upsilon, Greek-->
<!ENTITY phis     SDATA "[phis  ]"--/straightphi - straight phi-->
<!ENTITY Phi      SDATA "[Phi   ]"--=capital Phi, Greek-->
<!ENTITY phiv     SDATA "[phiv  ]"--/varphi - curly or open phi-->
<!ENTITY chi      SDATA "[chi   ]"--=small chi, Greek-->
<!ENTITY psi      SDATA "[psi   ]"--=small psi, Greek-->
<!ENTITY Psi      SDATA "[Psi   ]"--=capital Psi, Greek-->
<!ENTITY omega    SDATA "[omega ]"--=small omega, Greek-->
<!ENTITY Omega    SDATA "[Omega ]"--=capital Omega, Greek-->
```

**Exhibit 6-7** Entity set for Western European accented characters (iso-lat1.ent)

```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOlat1 PUBLIC
       "ISO 8879-1986//ENTITIES Added Latin 1//EN">
     %ISOlat1;
-->
<!ENTITY aacute SDATA "[aacute]"--=small a, acute accent-->
<!ENTITY Aacute SDATA "[Aacute]"--=capital A, acute accent-->
<!ENTITY acirc  SDATA "[acirc ]"--=small a, circumflex accent-->
<!ENTITY Acirc  SDATA "[Acirc ]"--=capital A, circumflex accent-->
<!ENTITY agrave SDATA "[agrave]"--=small a, grave accent-->
<!ENTITY Agrave SDATA "[Agrave]"--=capital A, grave accent-->
<!ENTITY aring  SDATA "[aring ]"--=small a, ring-->
<!ENTITY Aring  SDATA "[Aring ]"--=capital A, ring-->
<!ENTITY atilde SDATA "[atilde]"--=small a, tilde-->
<!ENTITY Atilde SDATA "[Atilde]"--=capital A, tilde-->
<!ENTITY auml   SDATA "[auml  ]"--=small a, dieresis or umlaut mark-->
<!ENTITY Auml   SDATA "[Auml  ]"--=capital A, dieresis or umlaut mark-->
<!ENTITY aelig  SDATA "[aelig ]"--=small ae diphthong (ligature)-->
<!ENTITY AElig  SDATA "[AElig ]"--=capital AE diphthong (ligature)-->
<!ENTITY ccedil SDATA "[ccedil]"--=small c, cedilla-->
<!ENTITY Ccedil SDATA "[Ccedil]"--=capital C, cedilla-->
<!ENTITY eth    SDATA "[eth   ]"--=small eth, Icelandic-->
<!ENTITY ETH    SDATA "[ETH   ]"--=capital Eth, Icelandic-->
<!ENTITY eacute SDATA "[eacute]"--=small e, acute accent-->
<!ENTITY Eacute SDATA "[Eacute]"--=capital E, acute accent-->
<!ENTITY ecirc  SDATA "[ecirc ]"--=small e, circumflex accent-->
<!ENTITY Ecirc  SDATA "[Ecirc ]"--=capital E, circumflex accent-->
<!ENTITY egrave SDATA "[egrave]"--=small e, grave accent-->
<!ENTITY Egrave SDATA "[Egrave]"--=capital E, grave accent-->
<!ENTITY euml   SDATA "[euml  ]"--=small e, dieresis or umlaut mark-->
<!ENTITY Euml   SDATA "[Euml  ]"--=capital E, dieresis or umlaut mark-->
<!ENTITY iacute SDATA "[iacute]"--=small i, acute accent-->
<!ENTITY Iacute SDATA "[Iacute]"--=capital I, acute accent-->
<!ENTITY icirc  SDATA "[icirc ]"--=small i, circumflex accent-->
<!ENTITY Icirc  SDATA "[Icirc ]"--=capital I, circumflex accent-->
<!ENTITY igrave SDATA "[igrave]"--=small i, grave accent-->
<!ENTITY Igrave SDATA "[Igrave]"--=capital I, grave accent-->
<!ENTITY iuml   SDATA "[iuml  ]"--=small i, dieresis or umlaut mark-->
<!ENTITY Iuml   SDATA "[Iuml  ]"--=capital I, dieresis or umlaut mark-->
<!ENTITY ntilde SDATA "[ntilde]"--=small n, tilde-->
<!ENTITY Ntilde SDATA "[Ntilde]"--=capital N, tilde-->
<!ENTITY oacute SDATA "[oacute]"--=small o, acute accent-->
<!ENTITY Oacute SDATA "[Oacute]"--=capital O, acute accent-->
<!ENTITY ocirc  SDATA "[ocirc ]"--=small o, circumflex accent-->
<!ENTITY Ocirc  SDATA "[Ocirc ]"--=capital O, circumflex accent-->
```

```
<!ENTITY ograve SDATA "[ograve]"--=small o, grave accent-->
<!ENTITY Ograve SDATA "[Ograve]"--=capital O, grave accent-->
<!ENTITY oslash SDATA "[oslash]"--=small o, slash-->
<!ENTITY Oslash SDATA "[Oslash]"--=capital O, slash-->
<!ENTITY otilde SDATA "[otilde]"--=small o, tilde-->
<!ENTITY Otilde SDATA "[Otilde]"--=capital O, tilde-->
<!ENTITY ouml   SDATA "[ouml  ]"--=small o, dieresis or umlaut mark-->
<!ENTITY Ouml   SDATA "[Ouml  ]"--=capital O, dieresis or umlaut mark-->
<!ENTITY szlig  SDATA "[szlig ]"--=small sharp s, German (sz ligature)-->
<!ENTITY thorn  SDATA "[thorn ]"--=small thorn, Icelandic-->
<!ENTITY THORN  SDATA "[THORN ]"--=capital THORN, Icelandic-->
<!ENTITY uacute SDATA "[uacute]"--=small u, acute accent-->
<!ENTITY Uacute SDATA "[Uacute]"--=capital U, acute accent-->
<!ENTITY ucirc  SDATA "[ucirc ]"--=small u, circumflex accent-->
<!ENTITY Ucirc  SDATA "[Ucirc ]"--=capital U, circumflex accent-->
<!ENTITY ugrave SDATA "[ugrave]"--=small u, grave accent-->
<!ENTITY Ugrave SDATA "[Ugrave]"--=capital U, grave accent-->
<!ENTITY uuml   SDATA "[uuml  ]"--=small u, dieresis or umlaut mark-->
<!ENTITY Uuml   SDATA "[Uuml  ]"--=capital U, dieresis or umlaut mark-->
<!ENTITY yacute SDATA "[yacute]"--=small y, acute accent-->
<!ENTITY Yacute SDATA "[Yacute]"--=capital Y, acute accent-->
<!ENTITY yuml   SDATA "[yuml  ]"--=small y, dieresis or umlaut mark-->
```

**Exhibit 6-8** Entity set for Eastern European accented characters (iso-lat2.ent)

```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOlat2 PUBLIC
       "ISO 8879-1986//ENTITIES Added Latin 2//EN">
     %ISOlat2;
-->
<!ENTITY abreve SDATA "[abreve]"--=small a, breve-->
<!ENTITY Abreve SDATA "[Abreve]"--=capital A, breve-->
<!ENTITY amacr  SDATA "[amacr ]"--=small a, macron-->
<!ENTITY Amacr  SDATA "[Amacr ]"--=capital A, macron-->
<!ENTITY aogon  SDATA "[aogon ]"--=small a, ogonek-->
<!ENTITY Aogon  SDATA "[Aogon ]"--=capital A, ogonek-->
<!ENTITY cacute SDATA "[cacute]"--=small c, acute accent-->
<!ENTITY Cacute SDATA "[Cacute]"--=capital C, acute accent-->
<!ENTITY ccaron SDATA "[ccaron]"--=small c, caron-->
<!ENTITY Ccaron SDATA "[Ccaron]"--=capital C, caron-->
<!ENTITY ccirc  SDATA "[ccirc ]"--=small c, circumflex accent-->
<!ENTITY Ccirc  SDATA "[Ccirc ]"--=capital C, circumflex accent-->
<!ENTITY cdot   SDATA "[cdot  ]"--=small c, dot above-->
<!ENTITY Cdot   SDATA "[Cdot  ]"--=capital C, dot above-->
<!ENTITY dcaron SDATA "[dcaron]"--=small d, caron-->
<!ENTITY Dcaron SDATA "[Dcaron]"--=capital D, caron-->
<!ENTITY dstrok SDATA "[dstrok]"--=small d, stroke-->
<!ENTITY Dstrok SDATA "[Dstrok]"--=capital D, stroke-->
<!ENTITY ecaron SDATA "[ecaron]"--=small e, caron-->
<!ENTITY Ecaron SDATA "[Ecaron]"--=capital E, caron-->
<!ENTITY edot   SDATA "[edot  ]"--=small e, dot above-->
<!ENTITY Edot   SDATA "[Edot  ]"--=capital E, dot above-->
<!ENTITY emacr  SDATA "[emacr ]"--=small e, macron-->
<!ENTITY Emacr  SDATA "[Emacr ]"--=capital E, macron-->
<!ENTITY eogon  SDATA "[eogon ]"--=small e, ogonek-->
<!ENTITY Eogon  SDATA "[Eogon ]"--=capital E, ogonek-->
<!ENTITY gacute SDATA "[gacute]"--=small g, acute accent-->
```

```
<!ENTITY gbreve SDATA "[gbreve]"--=small g, breve-->
<!ENTITY Gbreve SDATA "[Gbreve]"--=capital G, breve-->
<!ENTITY Gcedil SDATA "[Gcedil]"--=capital G, cedilla-->
<!ENTITY gcirc  SDATA "[gcirc ]"--=small g, circumflex accent-->
<!ENTITY Gcirc  SDATA "[Gcirc ]"--=capital G, circumflex accent-->
<!ENTITY gdot   SDATA "[gdot  ]"--=small g, dot above-->
<!ENTITY Gdot   SDATA "[Gdot  ]"--=capital G, dot above-->
<!ENTITY hcirc  SDATA "[hcirc ]"--=small h, circumflex accent-->
<!ENTITY Hcirc  SDATA "[Hcirc ]"--=capital H, circumflex accent-->
<!ENTITY hstrok SDATA "[hstrok]"--=small h, stroke-->
<!ENTITY Hstrok SDATA "[Hstrok]"--=capital H, stroke-->
<!ENTITY Idot   SDATA "[Idot  ]"--=capital I, dot above-->
<!ENTITY Imacr  SDATA "[Imacr ]"--=capital I, macron-->
<!ENTITY imacr  SDATA "[imacr ]"--=small i, macron-->
<!ENTITY ijlig  SDATA "[ijlig ]"--=small ij ligature-->
<!ENTITY IJlig  SDATA "[IJlig ]"--=capital IJ ligature-->
<!ENTITY inodot SDATA "[inodot]"--=small i without dot-->
<!ENTITY iogon  SDATA "[iogon ]"--=small i, ogonek-->
<!ENTITY Iogon  SDATA "[Iogon ]"--=capital I, ogonek-->
<!ENTITY itilde SDATA "[itilde]"--=small i, tilde-->
<!ENTITY Itilde SDATA "[Itilde]"--=capital I, tilde-->
<!ENTITY jcirc  SDATA "[jcirc ]"--=small j, circumflex accent-->
<!ENTITY Jcirc  SDATA "[Jcirc ]"--=capital J, circumflex accent-->
<!ENTITY kcedil SDATA "[kcedil]"--=small k, cedilla-->
<!ENTITY Kcedil SDATA "[Kcedil]"--=capital K, cedilla-->
<!ENTITY kgreen SDATA "[kgreen]"--=small k, Greenlandic-->
<!ENTITY lacute SDATA "[lacute]"--=small l, acute accent-->
<!ENTITY Lacute SDATA "[Lacute]"--=capital L, acute accent-->
<!ENTITY lcaron SDATA "[lcaron]"--=small l, caron-->
<!ENTITY Lcaron SDATA "[Lcaron]"--=capital L, caron-->
<!ENTITY lcedil SDATA "[lcedil]"--=small l, cedilla-->
<!ENTITY Lcedil SDATA "[Lcedil]"--=capital L, cedilla-->
<!ENTITY lmidot SDATA "[lmidot]"--=small l, middle dot-->
<!ENTITY Lmidot SDATA "[Lmidot]"--=capital L, middle dot-->
<!ENTITY lstrok SDATA "[lstrok]"--=small l, stroke-->
<!ENTITY Lstrok SDATA "[Lstrok]"--=capital L, stroke-->
<!ENTITY nacute SDATA "[nacute]"--=small n, acute accent-->
<!ENTITY Nacute SDATA "[Nacute]"--=capital N, acute accent-->
<!ENTITY eng    SDATA "[eng   ]"--=small eng, Lapp-->
<!ENTITY ENG    SDATA "[ENG   ]"--=capital ENG, Lapp-->
<!ENTITY napos  SDATA "[napos ]"--=small n, apostrophe-->
<!ENTITY ncaron SDATA "[ncaron]"--=small n, caron-->
<!ENTITY Ncaron SDATA "[Ncaron]"--=capital N, caron-->
<!ENTITY ncedil SDATA "[ncedil]"--=small n, cedilla-->
<!ENTITY Ncedil SDATA "[Ncedil]"--=capital N, cedilla-->
<!ENTITY odblac SDATA "[odblac]"--=small o, double acute accent-->
<!ENTITY Odblac SDATA "[Odblac]"--=capital O, double acute accent-->
<!ENTITY Omacr  SDATA "[Omacr ]"--=capital O, macron-->
<!ENTITY omacr  SDATA "[omacr ]"--=small o, macron-->
<!ENTITY oelig  SDATA "[oelig ]"--=small oe ligature-->
<!ENTITY OElig  SDATA "[OElig ]"--=capital OE ligature-->
<!ENTITY racute SDATA "[racute]"--=small r, acute accent-->
<!ENTITY Racute SDATA "[Racute]"--=capital R, acute accent-->
<!ENTITY rcaron SDATA "[rcaron]"--=small r, caron-->
<!ENTITY Rcaron SDATA "[Rcaron]"--=capital R, caron-->
<!ENTITY rcedil SDATA "[rcedil]"--=small r, cedilla-->
<!ENTITY Rcedil SDATA "[Rcedil]"--=capital R, cedilla-->
<!ENTITY sacute SDATA "[sacute]"--=small s, acute accent-->
<!ENTITY Sacute SDATA "[Sacute]"--=capital S, acute accent-->
<!ENTITY scaron SDATA "[scaron]"--=small s, caron-->
<!ENTITY Scaron SDATA "[Scaron]"--=capital S, caron-->
<!ENTITY scedil SDATA "[scedil]"--=small s, cedilla-->
<!ENTITY Scedil SDATA "[Scedil]"--=capital S, cedilla-->
<!ENTITY scirc  SDATA "[scirc ]"--=small s, circumflex accent-->
```

```
<!ENTITY Scirc  SDATA "[Scirc ]"--=capital S, circumflex accent-->
<!ENTITY tcaron SDATA "[tcaron]"--=small t, caron-->
<!ENTITY Tcaron SDATA "[Tcaron]"--=capital T, caron-->
<!ENTITY tcedil SDATA "[tcedil]"--=small t, cedilla-->
<!ENTITY Tcedil SDATA "[Tcedil]"--=capital T, cedilla-->
<!ENTITY tstrok SDATA "[tstrok]"--=small t, stroke-->
<!ENTITY Tstrok SDATA "[Tstrok]"--=capital T, stroke-->
<!ENTITY ubreve SDATA "[ubreve]"--=small u, breve-->
<!ENTITY Ubreve SDATA "[Ubreve]"--=capital U, breve-->
<!ENTITY udblac SDATA "[udblac]"--=small u, double acute accent-->
<!ENTITY Udblac SDATA "[Udblac]"--=capital U, double acute accent-->
<!ENTITY umacr  SDATA "[umacr ]"--=small u, macron-->
<!ENTITY Umacr  SDATA "[Umacr ]"--=capital U, macron-->
<!ENTITY uogon  SDATA "[uogon ]"--=small u, ogonek-->
<!ENTITY Uogon  SDATA "[Uogon ]"--=capital U, ogonek-->
<!ENTITY uring  SDATA "[uring ]"--=small u, ring-->
<!ENTITY Uring  SDATA "[Uring ]"--=capital U, ring-->
<!ENTITY utilde SDATA "[utilde]"--=small u, tilde-->
<!ENTITY Utilde SDATA "[Utilde]"--=capital U, tilde-->
<!ENTITY wcirc  SDATA "[wcirc ]"--=small w, circumflex accent-->
<!ENTITY Wcirc  SDATA "[Wcirc ]"--=capital W, circumflex accent-->
<!ENTITY ycirc  SDATA "[ycirc ]"--=small y, circumflex accent-->
<!ENTITY Ycirc  SDATA "[Ycirc ]"--=capital Y, circumflex accent-->
<!ENTITY Yuml   SDATA "[Yuml  ]"--=capital Y, dieresis or umlaut mark-->
<!ENTITY zacute SDATA "[zacute]"--=small z, acute accent-->
<!ENTITY Zacute SDATA "[Zacute]"--=capital Z, acute accent-->
<!ENTITY zcaron SDATA "[zcaron]"--=small z, caron-->
<!ENTITY Zcaron SDATA "[Zcaron]"--=capital Z, caron-->
<!ENTITY zdot   SDATA "[zdot  ]"--=small z, dot above-->
<!ENTITY Zdot   SDATA "[Zdot  ]"--=capital Z, dot above-->
```

**Exhibit 6-9** Entity set for commercial characters (iso-num.ent)

```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOnum PUBLIC
       "ISO 8879-1986//ENTITIES Numeric and Special Graphic//EN">
     %ISOnum;
-->
<!ENTITY half   SDATA "[half  ]"--=fraction one-half-->
<!ENTITY frac12 SDATA "[frac12]"--=fraction one-half-->
<!ENTITY frac14 SDATA "[frac14]"--=fraction one-quarter-->
<!ENTITY frac34 SDATA "[frac34]"--=fraction three-quarters-->
<!ENTITY frac18 SDATA "[frac18]"--=fraction one-eighth-->
<!ENTITY frac38 SDATA "[frac38]"--=fraction three-eighths-->
<!ENTITY frac58 SDATA "[frac58]"--=fraction five-eighths-->
<!ENTITY frac78 SDATA "[frac78]"--=fraction seven-eighths-->

<!ENTITY sup1   SDATA "[sup1  ]"--=superscript one-->
<!ENTITY sup2   SDATA "[sup2  ]"--=superscript two-->
<!ENTITY sup3   SDATA "[sup3  ]"--=superscript three-->

<!ENTITY plus   SDATA "[plus  ]"--=plus sign B:-- >
<!ENTITY plusmn SDATA "[plusmn]"--/pm B: =plus-or-minus sign-->
<!ENTITY lt     SDATA "[lt    ]"--=less-than sign R:-->
<!ENTITY equals SDATA "[equals]"--=equals sign R:-->
<!ENTITY gt     SDATA "[gt    ]"--=greater-than sign R:-->
<!ENTITY divide SDATA "[divide]"--/div B: =divide sign-->
```

```
<!ENTITY times  SDATA "[times ]"--/times B: =multiply sign-->

<!ENTITY curren SDATA "[curren]"--=general currency sign-->
<!ENTITY pound  SDATA "[pound ]"--=pound sign-->
<!ENTITY dollar SDATA "[dollar]"--=dollar sign-->
<!ENTITY cent   SDATA "[cent  ]"--=cent sign-->
<!ENTITY yen    SDATA "[yen   ]"--/yen =yen sign-->

<!ENTITY num    SDATA "[num   ]"--=number sign-->
<!ENTITY percnt SDATA "[percnt]"--=percent sign-->
<!ENTITY amp    SDATA "[amp   ]"--=ampersand-->
<!ENTITY ast    SDATA "[ast   ]"--/ast B: =asterisk-->
<!ENTITY commat SDATA "[commat]"--=commercial at-->
<!ENTITY lsqb   SDATA "[lsqb  ]"--/lbrack O: =left square bracket-->
<!ENTITY bsol   SDATA "[bsol  ]"--/backslash =reverse solidus-->
<!ENTITY rsqb   SDATA "[rsqb  ]"--/rbrack C: =right square bracket-->
<!ENTITY lcub   SDATA "[lcub  ]"--/lbrace O: =left curly bracket-->
<!ENTITY horbar SDATA "[horbar]"--=horizontal bar-->
<!ENTITY verbar SDATA "[verbar]"--/vert =vertical bar-->
<!ENTITY rcub   SDATA "[rcub  ]"--/rbrace C: =right curly bracket-->
<!ENTITY micro  SDATA "[micro ]"--=micro sign-->
<!ENTITY ohm    SDATA "[ohm   ]"--=ohm sign-->
<!ENTITY deg    SDATA "[deg   ]"--=degree sign-->
<!ENTITY ordm   SDATA "[ordm  ]"--=ordinal indicator, masculine-->
<!ENTITY ordf   SDATA "[ordf  ]"--=ordinal indicator, feminine-->
<!ENTITY sect   SDATA "[sect  ]"--=section sign-->
<!ENTITY para   SDATA "[para  ]"--=pilcrow (paragraph sign)-->
<!ENTITY middot SDATA "[middot]"--/centerdot B: =middle dot-->
<!ENTITY larr   SDATA "[larr  ]"--/leftarrow /gets A: =leftward arrow-->
<!ENTITY rarr   SDATA "[rarr  ]"--/rightarrow /to A: =rightward arrow-->
<!ENTITY uarr   SDATA "[uarr  ]"--/uparrow A: =upward arrow-->
<!ENTITY darr   SDATA "[darr  ]"--/downarrow A: =downward arrow-->
<!ENTITY copy   SDATA "[copy  ]"--=copyright sign-->
<!ENTITY reg    SDATA "[reg   ]"--/circledR =registered sign-->
<!ENTITY trade  SDATA "[trade ]"--=trade mark sign-->
<!ENTITY brvbar SDATA "[brvbar]"--=broken (vertical) bar-->
<!ENTITY not    SDATA "[not   ]"--/neg /lnot =not sign-->
<!ENTITY sung   SDATA "[sung  ]"--=music note (sung text sign)-->

<!ENTITY excl   SDATA "[excl  ]"--=exclamation mark-->
<!ENTITY iexcl  SDATA "[iexcl ]"--=inverted exclamation mark-->
<!ENTITY quot   SDATA "[quot  ]"--=quotation mark-->
<!ENTITY apos   SDATA "[apos  ]"--=apostrophe-->
<!ENTITY lpar   SDATA "[lpar  ]"--O: =left parenthesis-->
<!ENTITY rpar   SDATA "[rpar  ]"--C: =right parenthesis-->
<!ENTITY comma  SDATA "[comma ]"--P: =comma-->
<!ENTITY lowbar SDATA "[lowbar]"--=low line-->
<!ENTITY hyphen SDATA "[hyphen]"--=hyphen-->
<!ENTITY period SDATA "[period]"--=full stop, period-->
<!ENTITY sol    SDATA "[sol   ]"--=solidus-->
<!ENTITY colon  SDATA "[colon ]"--/colon P:-->
<!ENTITY semi   SDATA "[semi  ]"--=semicolon P:-->
<!ENTITY quest  SDATA "[quest ]"--=question mark-->
<!ENTITY iquest SDATA "[iquest]"--=inverted question mark-->
<!ENTITY laquo  SDATA "[laquo ]"--=angle quotation mark, left-->
<!ENTITY raquo  SDATA "[raquo ]"--=angle quotation mark, right-->
<!ENTITY lsquo  SDATA "[lsquo ]"--=single quotation mark, left-->
<!ENTITY rsquo  SDATA "[rsquo ]"--=single quotation mark, right-->
<!ENTITY ldquo  SDATA "[ldquo ]"--=double quotation mark, left-->
<!ENTITY rdquo  SDATA "[rdquo ]"--=double quotation mark, right-->
<!ENTITY nbsp   SDATA "[nbsp  ]"--=no break (required) space-->
<!ENTITY shy    SDATA "[shy   ]"--=soft hyphen-->
```

## Exhibit 6-10 Entity set for publishers' special characters (iso-pub.ent)

```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOpub PUBLIC
       "ISO 8879-1986//ENTITIES Publishing//EN">
     %ISOpub;
-->
<!ENTITY emsp   SDATA "[emsp  ]"--=em space-->
<!ENTITY ensp   SDATA "[ensp  ]"--=en space (1/2-em)-->
<!ENTITY emsp13 SDATA "[emsp3 ]"--=1/3-em space-->
<!ENTITY emsp14 SDATA "[emsp4 ]"--=1/4-em space-->
<!ENTITY numsp  SDATA "[numsp ]"--=digit space (width of a number)-->
<!ENTITY puncsp SDATA "[puncsp]"--=punctuation space (width of comma)-->
<!ENTITY thinsp SDATA "[thinsp]"--=thin space (1/6-em)-->
<!ENTITY hairsp SDATA "[hairsp]"--=hair space-->
<!ENTITY mdash  SDATA "[mdash ]"--=em dash-->
<!ENTITY ndash  SDATA "[ndash ]"--=en dash-->
<!ENTITY dash   SDATA "[dash  ]"--=hyphen (true graphic)-->
<!ENTITY blank  SDATA "[blank ]"--=significant blank symbol-->
<!ENTITY hellip SDATA "[hellip]"--=ellipsis (horizontal)-->
<!ENTITY nldr   SDATA "[nldr  ]"--=double baseline dot (en leader)-->
<!ENTITY frac13 SDATA "[frac13]"--=fraction one-third-->
<!ENTITY frac23 SDATA "[frac23]"--=fraction two-thirds-->
<!ENTITY frac15 SDATA "[frac15]"--=fraction one-fifth-->
<!ENTITY frac25 SDATA "[frac25]"--=fraction two-fifths-->
<!ENTITY frac35 SDATA "[frac35]"--=fraction three-fifths-->
<!ENTITY frac45 SDATA "[frac45]"--=fraction four-fifths-->
<!ENTITY frac16 SDATA "[frac16]"--=fraction one-sixth-->
<!ENTITY frac56 SDATA "[frac56]"--=fraction five-sixths-->
<!ENTITY incare SDATA "[incare]"--=in-care-of symbol-->
<!ENTITY block  SDATA "[block ]"--=full block-->
<!ENTITY uhblk  SDATA "[uhblk ]"--=upper half block-->
<!ENTITY lhblk  SDATA "[lhblk ]"--=lower half block-->
<!ENTITY blk14  SDATA "[blk14 ]"--=25% shaded block-->
<!ENTITY blk12  SDATA "[blk12 ]"--=50% shaded block-->
<!ENTITY blk34  SDATA "[blk34 ]"--=75% shaded block-->
<!ENTITY marker SDATA "[marker]"--=histogram marker-->
<!ENTITY cir    SDATA "[cir   ]"--/circ B: =circle, open-->
<!ENTITY squ    SDATA "[squ   ]"--=square, open-->
<!ENTITY rect   SDATA "[rect  ]"--=rectangle, open-->
<!ENTITY utri   SDATA "[utri  ]"--/triangle =up triangle, open-->
<!ENTITY dtri   SDATA "[dtri  ]"--/triangledown =down triangle, open-->
<!ENTITY star   SDATA "[star  ]"--=star, open-->
<!ENTITY bull   SDATA "[bull  ]"--/bullet B: =round bullet, filled-->
<!ENTITY squf   SDATA "[squf  ]"--/blacksquare =sq bullet, filled-->
<!ENTITY utrif  SDATA "[utrif ]"--/blacktriangle =up tri, filled-->
<!ENTITY dtrif  SDATA "[dtrif ]"--/blacktriangledown =dn tri, filled-->
<!ENTITY ltrif  SDATA "[ltrif ]"--/blacktriangleleft R: =l tri, filled-->
<!ENTITY rtrif  SDATA "[rtrif ]"--/blacktriangleright R: =r tri, filled-->
<!ENTITY clubs  SDATA "[clubs ]"--/clubsuit =club suit symbol-->
<!ENTITY diams  SDATA "[diams ]"--/diamondsuit =diamond suit symbol-->
<!ENTITY hearts SDATA "[hearts]"--/heartsuit =heart suit symbol-->
<!ENTITY spades SDATA "[spades]"--/spadesuit =spades suit symbol-->
<!ENTITY malt   SDATA "[malt  ]"--/maltese =maltese cross-->
<!ENTITY dagger SDATA "[dagger]"--/dagger B: =dagger-->
<!ENTITY Dagger SDATA "[Dagger]"--/ddagger B: =double dagger-->
<!ENTITY check  SDATA "[check ]"--/checkmark =tick, check mark-->
<!ENTITY cross  SDATA "[ballot]"--=ballot cross-->
```

```
<!ENTITY sharp  SDATA "[sharp ]"--/sharp =musical sharp-->
<!ENTITY flat   SDATA "[flat  ]"--/flat =musical flat-->
<!ENTITY male   SDATA "[male  ]"--=male symbol-->
<!ENTITY female SDATA "[female]"--=female symbol-->
<!ENTITY phone  SDATA "[phone ]"--=telephone symbol-->
<!ENTITY telrec SDATA "[telrec]"--=telephone recorder symbol-->
<!ENTITY copysr SDATA "[copysr]"--=sound recording copyright sign-->
<!ENTITY caret  SDATA "[caret ]"--=caret (insertion mark)-->
<!ENTITY lsquor SDATA "[lsquor]"--=rising single quote, left (low)-->
<!ENTITY ldquor SDATA "[ldquor]"--=rising dbl quote, left (low)-->

<!ENTITY fflig  SDATA "[fflig ]"--small ff ligature-->
<!ENTITY filig  SDATA "[filig ]"--small fi ligature-->
<!ENTITY fjlig  SDATA "[fjlig ]"--small fj ligature-->
<!ENTITY ffilig SDATA "[ffilig]"--small ffi ligature-->
<!ENTITY ffllig SDATA "[ffllig]"--small ffl ligature-->
<!ENTITY fllig  SDATA "[fllig ]"--small fl ligature-->

<!ENTITY mldr   SDATA "[mldr  ]"--em leader-->
<!ENTITY rdquor SDATA "[rdquor]"--rising dbl quote, right (high)-->
<!ENTITY rsquor SDATA "[rsquor]"--rising single quote, right (high)-->
<!ENTITY vellip SDATA "[vellip]"--vertical ellipsis-->

<!ENTITY hybull SDATA "[hybull]"--rectangle, filled (hyphen bullet)-->
<!ENTITY loz    SDATA "[loz   ]"--/lozenge - lozenge or total mark-->
<!ENTITY lozf   SDATA "[lozf  ]"--/blacklozenge - lozenge, filled-->
<!ENTITY ltri   SDATA "[ltri  ]"--/triangleleft B: l triangle, open-->
<!ENTITY rtri   SDATA "[rtri  ]"--/triangleright B: r triangle, open-->
<!ENTITY starf  SDATA "[starf ]"--/bigstar - star, filled-->

<!ENTITY natur  SDATA "[natur ]"--/natural - music natural-->
<!ENTITY rx     SDATA "[rx    ]"--pharmaceutical prescription (Rx)-->
<!ENTITY sext   SDATA "[sext  ]"--sextile (6-pointed star)-->

<!ENTITY target SDATA "[target]"--register mark or target-->
<!ENTITY dlcrop SDATA "[dlcrop]"--downward left crop mark -->
<!ENTITY drcrop SDATA "[drcrop]"--downward right crop mark -->
<!ENTITY ulcrop SDATA "[ulcrop]"--upward left crop mark -->
<!ENTITY urcrop SDATA "[urcrop]"--upward right crop mark -->
```

**Exhibit 6-11** Entity set for technical special characters (iso-tech.ent)

```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOtech PUBLIC
       "ISO 8879-1986//ENTITIES General Technical//EN">
     %ISOtech;
-->
<!ENTITY aleph  SDATA "[aleph ]"--/aleph =aleph, Hebrew-->
<!ENTITY and    SDATA "[and   ]"--/wedge /land B: =logical and-->
<!ENTITY ang90  SDATA "[ang90 ]"--=right (90 degree) angle-->
<!ENTITY angsph SDATA "[angsph]"--/sphericalangle =angle-spherical-->
<!ENTITY ap     SDATA "[ap    ]"--/approx R: =approximate-->
<!ENTITY becaus SDATA "[becaus]"--/because R: =because-->
<!ENTITY bottom SDATA "[bottom]"--/bot B: =perpendicular-->
<!ENTITY cap    SDATA "[cap   ]"--/cap B: =intersection-->
<!ENTITY cong   SDATA "[cong  ]"--/cong R: =congruent with-->
<!ENTITY conint SDATA "[conint]"--/oint L: =contour integral operator-->
```

```
<!ENTITY cup    SDATA "[cup   ]"--/cup B: =union or logical sum-->
<!ENTITY equiv  SDATA "[equiv ]"--/equiv R: =identical with-->
<!ENTITY exist  SDATA "[exist ]"--/exists =at least one exists-->
<!ENTITY forall SDATA "[forall]"--/forall =for all-->
<!ENTITY fnof   SDATA "[fnof  ]"--=function of (italic small f)-->
<!ENTITY ge     SDATA "[ge    ]"--/geq /ge R: =greater-than-or-equal-->
<!ENTITY iff    SDATA "[iff   ]"--/iff =if and only if-->
<!ENTITY infin  SDATA "[infin ]"--/infty =infinity-->
<!ENTITY int    SDATA "[int   ]"--/int L: =integral operator-->
<!ENTITY isin   SDATA "[isin  ]"--/in R: =set membership-->
<!ENTITY lang   SDATA "[lang  ]"--/langle O: =left angle bracket-->
<!ENTITY lArr   SDATA "[lArr  ]"--/Leftarrow A: =is implied by-->
<!ENTITY le     SDATA "[le    ]"--/leq /le R: =less-than-or-equal-->
<!ENTITY minus  SDATA "[minus ]"--B: =minus sign-->
<!ENTITY mnplus SDATA "[mnplus]"--/mp B: =minus-or-plus sign-->
<!ENTITY nabla  SDATA "[nabla ]"--/nabla =del, Hamilton operator-->
<!ENTITY ne     SDATA "[ne    ]"--/ne /neq R: =not equal-->
<!ENTITY ni     SDATA "[ni    ]"--/ni /owns R: =contains-->
<!ENTITY or     SDATA "[or    ]"--/vee /lor B: =logical or-->
<!ENTITY par    SDATA "[par   ]"--/parallel R: =parallel-->
<!ENTITY part   SDATA "[part  ]"--/partial =partial differential-->
<!ENTITY permil SDATA "[permil]"--=per thousand-->
<!ENTITY perp   SDATA "[perp  ]"--/perp R: =perpendicular-->
<!ENTITY prime  SDATA "[prime ]"--/prime =prime or minute-->
<!ENTITY Prime  SDATA "[Prime ]"--=double prime or second-->
<!ENTITY prop   SDATA "[prop  ]"--/propto R: =is proportional to-->
<!ENTITY radic  SDATA "[radic ]"--/surd =radical-->
<!ENTITY rang   SDATA "[rang  ]"--/rangle C: =right angle bracket-->
<!ENTITY rArr   SDATA "[rArr  ]"--/Rightarrow A: =implies-->
<!ENTITY sim    SDATA "[sim   ]"--/sim R: =similar-->
<!ENTITY sime   SDATA "[sime  ]"--/simeq R: =similar, equals-->
<!ENTITY square SDATA "[square]"--/square B: =square-->
<!ENTITY sub    SDATA "[sub   ]"--/subset R: =subset or is implied by-->
<!ENTITY sube   SDATA "[sube  ]"--/subseteq R: =subset, equals-->
<!ENTITY sup    SDATA "[sup   ]"--/supset R: =superset or implies-->
<!ENTITY supe   SDATA "[supe  ]"--/supseteq R: =superset, equals-->
<!ENTITY there4 SDATA "[there4]"--/therefore R: =therefore-->
<!ENTITY Verbar SDATA "[Verbar]"--/Vert =dbl vertical bar-->

<!ENTITY angst  SDATA "[angst ]"--Angstrom =capital A, ring-->
<!ENTITY bernou SDATA "[bernou]"--Bernoulli function (script capital B)-->
<!ENTITY compfn SDATA "[compfn]"--B: composite function (small circle)-->
<!ENTITY Dot    SDATA "[Dot   ]"--=dieresis or umlaut mark-->
<!ENTITY DotDot SDATA "[DotDot]"--four dots above-->
<!ENTITY hamilt SDATA "[hamilt]"--Hamiltonian (script capital H)-->
<!ENTITY lagran SDATA "[lagran]"--Lagrangian (script capital L)-->
<!ENTITY lowast SDATA "[lowast]"--low asterisk-->
<!ENTITY notin  SDATA "[notin ]"--N: negated set membership-->
<!ENTITY order  SDATA "[order ]"--order of (script small o)-->
<!ENTITY phmmat SDATA "[phmmat]"--physics M-matrix (script capital M)-->
<!ENTITY tdot   SDATA "[tdot  ]"--three dots above-->
<!ENTITY tprime SDATA "[tprime]"--triple prime-->
<!ENTITY wedgeq SDATA "[wedgeq]"--R: corresponds to (wedge, equals)-->
```

**Exhibit 6-12** Entity set for keyboard characters (tcif_kb.ent)

```
<!-- Keybd-2 - TCIF Entity-reference set for keycaps - standard.
     Release: 2.4, 1997-07-16 (tcif_kb.ent) -->
<!-- Copyright (c) 1995-1997, Alliance for Telecommunications Industry
     Solutions (ATIS). Permission is hereby granted to use, copy, modify and
     distribute this material for any purpose and without fee, provided that
     this notice continue to appear in all copies.
```

```
     Reproduction and distribution for resale is prohibited. -->
<!-- Keycaps character entity set. Typical invocation:
     <!ENTITY % TCIFkb PUBLIC "-//USA/TCIF//ENTITIES Keybd-2//EN">
     %TCIFkb; -->
<!ENTITY newline      SDATA "[newline]"      --=newline-->
<!ENTITY smark        SDATA "[smark ]"       --=service mark (SM)-->
<!ENTITY hyph         SDATA "[hyph]"         --=literal hyphen-->
<!ENTITY again-key    SDATA "[again-key]"    --=again key-->
<!ENTITY alt-key      SDATA "[alt-key]"      --=alt key-->
<!ENTITY altg-key     SDATA "[altg-key]"     --=alt graph key-->
<!ENTITY amp-key      SDATA "[amp-key]"      --=ampersand key-->
<!ENTITY apple-key    SDATA "[apple-key]"    --=apple key-->
<!ENTITY break-key    SDATA "[break-key]"    --=break key-->
<!ENTITY bs-key       SDATA "[bs-key]"       --=backspace key-->
<!ENTITY caps-key     SDATA "[caps-key]"     --=capslock key-->
<!ENTITY clear-key    SDATA "[clear-key]"    --=clear key-->
<!ENTITY clover-key   SDATA "[clover-key]"   --=cloverleaf (option) key-->
<!ENTITY cmd-key      SDATA "[cmd-key]"      --=command key-->
<!ENTITY color-key    SDATA "[color-key]"    --=solid-color blank key-->
<!ENTITY compose-key  SDATA "[compose-key]"  --=compose [character] key-->
<!ENTITY copy-key     SDATA "[copy-key]"     --=copy key-->
<!ENTITY cr-key       SDATA "[cr-key]"       --=carriage return key-->
<!ENTITY ctrl-key     SDATA "[ctrl-key]"     --=control key-->
<!ENTITY cut-key      SDATA "[cut-key]"      --=cut key-->
<!ENTITY del-key      SDATA "[del-key]"      --=delete key-->
<!ENTITY delbk-arrow  SDATA "[delbk-arrow]"  --=delete-backward-arrow key-->
<!ENTITY delfd-arrow  SDATA "[delfd-arrow]"  --=delete-forward-arrow key-->
<!ENTITY dnlft-arrow  SDATA "[dnlft-arrow]"  --=down-left-arrow key-->
<!ENTITY dnrt-arrow   SDATA "[dnrt-arrow]"   --=down-right-arrow key-->
<!ENTITY do-key       SDATA "[do-key]"       --=do key-->
<!ENTITY down-arrow   SDATA "[down-arrow]"   --=down-arrow key-->
<!ENTITY eight-key    SDATA "[eight-key]"    --=8-* key-->
<!ENTITY end-key      SDATA "[end-key]"      --=end key-->
<!ENTITY enter-key    SDATA "[enter-key]"    --=enter key-->
<!ENTITY esc-key      SDATA "[esc-key]"      --=escape key-->
<!ENTITY f1-key       SDATA "[f1-key]"       --=function key 1-->
<!ENTITY f2-key       SDATA "[f2-key]"       --=function key 2-->
<!ENTITY f3-key       SDATA "[f3-key]"       --=function key 3-->
<!ENTITY f4-key       SDATA "[f4-key]"       --=function key 4-->
<!ENTITY f5-key       SDATA "[f5-key]"       --=function key 5-->
<!ENTITY f6-key       SDATA "[f6-key]"       --=function key 6-->
<!ENTITY f7-key       SDATA "[f7-key]"       --=function key 7-->
<!ENTITY f8-key       SDATA "[f8-key]"       --=function key 8-->
<!ENTITY f9-key       SDATA "[f9-key]"       --=function key 9-->
<!ENTITY f10-key      SDATA "[f10-key]"      --=function key 10-->
<!ENTITY f11-key      SDATA "[f11-key]"      --=function key 11-->
<!ENTITY f12-key      SDATA "[f12-key]"      --=function key 12-->
<!ENTITY f13-key      SDATA "[f13-key]"      --=function key 13-->
<!ENTITY f14-key      SDATA "[f14-key]"      --=function key 14-->
<!ENTITY f15-key      SDATA "[f15-key]"      --=function key 15-->
<!ENTITY f16-key      SDATA "[f16-key]"      --=function key 16-->
<!ENTITY f17-key      SDATA "[f17-key]"      --=function key 17-->
<!ENTITY f18-key      SDATA "[f18-key]"      --=function key 18-->
<!ENTITY f19-key      SDATA "[f19-key]"      --=function key 19-->
<!ENTITY f20-key      SDATA "[f20-key]"      --=function key 20-->
<!ENTITY f21-key      SDATA "[f21-key]"      --=function key 21-->
<!ENTITY f22-key      SDATA "[f22-key]"      --=function key 22-->
<!ENTITY f23-key      SDATA "[f23-key]"      --=function key 23-->
<!ENTITY f24-key      SDATA "[f24-key]"      --=function key 24-->
<!ENTITY find-key     SDATA "[find-key]"     --=find key-->
<!ENTITY five-key     SDATA "[five-key]"     --=5-% key-->
<!ENTITY four-key     SDATA "[four-key]"     --=4-$ key-->
<!ENTITY front-key    SDATA "[front-key]"    --=front key-->
<!ENTITY funct-key    SDATA "[funct-key;"    --=function key-->
```

```
<!ENTITY help-key     SDATA "[help-key]"     --=help key-->
<!ENTITY hold-key     SDATA "[hold-key]"     --=hold key-->
<!ENTITY home-key     SDATA "[home-key]"     --=home key-->
<!ENTITY ins-key      SDATA "[ins-key]"      --=insert key-->
<!ENTITY left-arrow   SDATA "[left-arrow]"   --=left-arrow key-->
<!ENTITY lf-key       SDATA "[lf-key]"       --=linefeed key-->
<!ENTITY lt-key       SDATA "[lt-key]"       --=less-than key-->
<!ENTITY macro-key    SDATA "[macro-key]"    --=macro key-->
<!ENTITY meta-key     SDATA "[meta-key]"     --=meta key-->
<!ENTITY next-key     SDATA "[next-key]"     --=next key-->
<!ENTITY nine-key     SDATA "[nine-key]"     --=9-( key-->
<!ENTITY numlk-key    SDATA "[numlk-key]"    --=numlock key-->
<!ENTITY onoff-key    SDATA "[onoff-key]"    --=on/off key-->
<!ENTITY one-key      SDATA "[one-key]"      --=1-! key-->
<!ENTITY open-key     SDATA "[open-key]"     --=open key-->
<!ENTITY opt-key      SDATA "[opt-key]"      --=option key-->
<!ENTITY paste-key    SDATA "[paste-key]"    --=paste key-->
<!ENTITY pause-key    SDATA "[pause-key]"    --=pause key-->
<!ENTITY pf1-key      SDATA "[pf1-key]"      --=function key 1-->
<!ENTITY pf2-key      SDATA "[pf2-key]"      --=function key 2-->
<!ENTITY pf3-key      SDATA "[pf3-key]"      --=function key 3-->
<!ENTITY pf4-key      SDATA "[pf4-key]"      --=function key 4-->
<!ENTITY pf5-key      SDATA "[pf5-key]"      --=function key 5-->
<!ENTITY pf6-key      SDATA "[pf6-key]"      --=function key 6-->
<!ENTITY pf7-key      SDATA "[pf7-key]"      --=function key 7-->
<!ENTITY pf8-key      SDATA "[pf8-key]"      --=function key 8-->
<!ENTITY pf9-key      SDATA "[pf9-key]"      --=function key 9-->
<!ENTITY pf10-key     SDATA "[pf10-key]"     --=function key 10-->
<!ENTITY pf11-key     SDATA "[pf11-key]"     --=function key 11-->
<!ENTITY pf12-key     SDATA "[pf12-key]"     --=function key 12-->
<!ENTITY pf13-key     SDATA "[pf13-key]"     --=function key 13-->
<!ENTITY pf14-key     SDATA "[pf14-key]"     --=function key 14-->
<!ENTITY pf15-key     SDATA "[pf15-key]"     --=function key 15-->
<!ENTITY pf16-key     SDATA "[pf16-key]"     --=function key 16-->
<!ENTITY pf17-key     SDATA "[pf17-key]"     --=function key 17-->
<!ENTITY pf18-key     SDATA "[pf18-key]"     --=function key 18-->
<!ENTITY pf19-key     SDATA "[pf19-key]"     --=function key 19-->
<!ENTITY pf20-key     SDATA "[pf20-key]"     --=function key 20-->
<!ENTITY pf21-key     SDATA "[pf21-key]"     --=function key 21-->
<!ENTITY pf22-key     SDATA "[pf22-key]"     --=function key 22-->
<!ENTITY pf23-key     SDATA "[pf23-key]"     --=function key 23-->
<!ENTITY pf24-key     SDATA "[pf24-key]"     --=function key 24-->
<!ENTITY pgdn-key     SDATA "[pgdn-key]"     --=page down key-->
<!ENTITY pgup-key     SDATA "[pgup-key]"     --=page up key-->
<!ENTITY plain-key    SDATA "[plain-key]"    --=plain blank key-->
<!ENTITY prev-key     SDATA "[prev-key]"     --=prev key-->
<!ENTITY print-key    SDATA "[print-key]"    --=print key-->
<!ENTITY props-key    SDATA "[props-key]"    --=props key-->
<!ENTITY prtscr-key   SDATA "[prtscr-key]"   --=print screen key-->
<!ENTITY ret-arrow    SDATA "[ret-arrow]"    --=return-arrow key-->
<!ENTITY right-arrow  SDATA "[right-arrow]"  --=right-arrow key-->
<!ENTITY scrlk-key    SDATA "[scrlk-key]"    --=scroll lock key-->
<!ENTITY sel-key      SDATA "[sel-key]"      --=select key-->
<!ENTITY setup-key    SDATA "[setup-key]"    --=setup key-->
<!ENTITY seven-key    SDATA "[seven-key]"    --=7-& key-->
<!ENTITY shift-key    SDATA "[shift-key]"    --=shift key-->
<!ENTITY six-key      SDATA "[six-key]"      --=6-^ key-->
<!ENTITY space-key    SDATA "[space-key]"    --=spacebar-->
<!ENTITY stop-key     SDATA "[stop-key]"     --=stop key-->
<!ENTITY sysrq-key    SDATA "[sysrq-key]"    --=sysrq key-->
<!ENTITY tab-key      SDATA "[tab-key]"      --=tab key-->
<!ENTITY tab-arrow    SDATA "[tab-arrow]"    --=tab-arrow key-->
<!ENTITY three-key    SDATA "[three-key]"    --=3-# key-->
<!ENTITY two-key      SDATA "[two-key]"      --=2-@ key-->
```

```
<!ENTITY undo-key     SDATA "[undo-key]"     --=undo key-->
<!ENTITY uplft-arrow  SDATA "[uplft-arrow]"  --=up-left-arrow key-->
<!ENTITY up-arrow     SDATA "[up-arrow]"     --=up-arrow key-->
<!ENTITY uprt-arrow   SDATA "[uprt-arrow]"   --=up-right-arrow key-->
<!ENTITY window-key   SDATA "[window-key]"   --=Windows-symbol key-->
<!ENTITY zero-key     SDATA "[zero-key]"     --=0-) key-->
```