Guide to the DocBook DTD

HaL Computer Systems International, Ltd.

and

O'Reilly & Associates, Inc.

DTD release 1.0
Documentation version 1.0 for release 1.0
10 November 1992

The DocBook DTD is maintained by O'Reilly & Associates. Please direct all questions, bug reports, or suggestions for changes to dtd@ora.com or by mail to: Terry Allen, O'Reilly & Associates, Inc., 103A Morris Street, Sebastopol, California, 95472.

Table of Contents

Acknowledgements

The DocBook DTD was developed by the joint efforts of Hal Computer Systems International, Ltd., and O'Reilly & Associates, Inc. Conleth O'Connell, Steve Williams, and David Bass of HaL Computer Systems and Terry Allen and Dale Dougherty of O'Reilly & Associates were the main contributors to the DTD and its documentation. Con deserves special mention as the chief architect of this DTD. A lot of other people at both companies contributed their insight and encouragement.

How to Get the DocBook DTD Online

The DocBook DTD and its documentation can be found online in the Davenport archive (/pub/davenport/docbook) at ftp.ora.com (140.186.65.25).
Here is a sample session showing you how to retrieve the DTD and its documentation:

```
% ftp ftp.ora.com
Connected to ruby.ora.com.
ruby FTP server (Version 6.12 Thu Apr 23 21:09:30 EDT 1992) ready.
Name (ftp.ora.com:dale): anonymous
Guest login ok, send e-mail address as password.
Password: <-- type e-mail address
ftp> cd pub/davenport/docbook
```

The DocBook DTD itself is in a file named "docbook.dtd":

```
ftp> get docbook.dtd
```

The "get" command will put this ASCII file on your system.
The DocBook documentation is a compressed PostScript file:

```
ftp> binary
ftp> get docbook.ps.Z
```

This will put the binary file docbook.ps.Z on your system. You must uncompress that file before sending it to a PostScript printer.

Goals in Developing the DocBook DTD

The DocBook DTD defines structural SGML markup for computer documentation and technical books.

Our primary goal in developing this DTD was to provide for the migration to SGML of existing computer documentation and technical books. It is a DTD that describes the kinds of structures we encountered in processing several large bodies of software documentation and technical books.

We wanted to be able to filter these documents into an SGML form from a variety of other formats, such as FrameMaker, troff, and RTF. Generally, each of the significant components is contained in a single file, so we designed the DTD so that any of these components (a chapter or a reference page) can be dealt with in isolation from others. For this reason, too, we treated title pages, tables of contents, glossaries, indexes, and so forth as individual components, not grouping them in front and back matter sections, as is commonly done. Within a book component, sectioning of the information usually begins with the book component itself, Chapter, for example. This section is broken up into smaller pieces of properly nested sections denoted by HeadX (Head1–Head5). Each of the sections is composed of the title information followed by some required content. That content is either paragraph-level components followed by lesser sections, or just lesser sections. Paragraph-level components form a group of elements that can occur any number of times and in any order. This is the level at which different documents begin to vary widely in how they represent their contents.

This DTD is descriptive of various forms rather than prescriptive of a preferred form. To make this distinction clear, let's look at the case of man pages, where there are many different forms of what is basically a fairly simple structure. We could have taken the approach that we wanted to construct an "ideal" man page in our DTD, but a great number of existing man pages would not fit into the ideal form. Thus, it would become an extra burden on a conversion effort to resolve different practices. We sought to accommodate the differences in a fairly open-ended manner. Nonetheless, if one planned to create new man pages, there is benefit in being more prescriptive. And one could do a variant of man page form used in this DTD (RefEntry) that indeed specified a set of elements in a very specific order.

We do not pretend that this DTD is all-encompassing. We took a pragmatic, bottom-up approach in designing the DTD. We sought to keep the DTD simple. Again, our motivation was to get the information into SGML so that we might better understand the impact of SGML on the production of documentation and technical books, in print and online. We believe strongly that getting real-world experience applying SGML is critical to understanding it.

Planned Revisions

This DTD already reflects the requirements of a diverse set of documents, and we expect that it will evolve as it is applied to more sources of computer documentation and technical books. Indeed, we expect that groups that begin to use the DTD will find it necessary to extend the DTD for their domain of applications. Anyone making such changes should read the section (provided by Con) titled "Notes on Modifying the DTD." We will continue to improve the DTD to make it more modular in design, and thus make it easier to accommodate local variations.

We are committed to setting up a review process by which the revisions made to the DTD by various groups can be evaluated for inclusion into the next release of the DTD. We will be sensitive to the fact that as of Release 1.0 of the DTD, there are SGML documents that depend on this DTD and those documents would have to be modified to reflect a new release of the DTD. However, the primary goal of the Release 1.0 DTD is to provide for the migration of documents into SGML; we expect subsequent releases of the DTD to reflect our growing experience with SGML and a greater understanding of its capabilities. We also believe that tools can be written that will aid considerably in migrating an SGML document from Release 1.0 to conform to any subsequent release.

We intend to follow the efforts of the Davenport Group, and support the recommendations of the Davenport Advisory Standard on Hypermedia (DASH) and the Committee for Common Man (CFCM). We anticipate changes in the following areas:

- the use of HyTime to represent link information (affecting the Link and XRef elements mainly)
- the use of SGML architectural forms to represent the structure of navigational documents, such as ToC, Index and Glossary
- the use of SGML architectural forms to make interchange of documents using different DTDs possible
- the use of HyTime to represent a wide array of hypermedia/multimedia objects to accommodate interactive audio and video presentations
- The RefEntry (man page) form may be modified. Its current form has been designed to represent existing reference man pages, and to be easily extensible toward supporting the effortss of the Committee for Common Man.

Our goal is to have this DTD reflect industry-accepted standards.

Tables are a complex problem. While one can model the rows and cells of a table as we have done, successful presentation of the table remains highly application-dependent. Some applications are tightly coupled to a specific Table DTD. We anticipate studying a range of applications and examining alternative approaches.

Conditional text is another area we believe is important to consider, but it has not been handled in the current release. Many of us have documents that vary slightly from one platform or interface to another, and they are difficult to maintain.

The information about a manual or book that in print typically appears on the title page is probably inadequately handled as subelements of TitlePage. We expect to review it for a future release. Its current use reflects information that is present in existing files.

Please send any suggestions for changes to dtd@ora.com or by mail to: Terry Allen, O'Reilly & Associates, Inc., 103A Morris Street, Sebastopol, California, 95472.

Using a DTD

A DTD (Document Type Definition) is a set of rules for describing the structure of a document in SGML (Standard Generalized Markup Language, ISO 8879) and for marking it up for electronic publication and transfer. An SGML document instance is a document marked up according to a particular DTD.

A DTD defines elements, attributes, entities, and their relationships. Elements are structures identified by markup in the document. Normally, an element is marked by begin and end tags:

```
<CHAPTER>
    <TITLE> X Window System User's Guide
    </TITLE>
    .
    .
    .
</CHAPTER>
```

Elements may have attributes, which are included within the start tag (<ITEMIZEDLIST MARK="Bullet">). Entities are short strings marked off with ampersands and semicolons, such as `&`. In brief:

> **Element.** A structural building block that can be defined to contain data, subordinate elements, or both. In typical documents, examples of elements would be paragraphs, chapters, headings, figures, and tables. In hypermedia documents, some examples are hyperlinks of various types, event schedules, events, and objects encoded in a variety of data notations.
>
> **Attribute.** A property, associated with elements of a given type, the value of which describes the element but is not part of its content (for example, the editor of a chapter or revision date of a table). Many element types have an ID attribute, which provides a label for each element of that type so that it can be referenced explicitly from any place in the document. The references are made by elements that have an ID reference attribute the value of which is the label of the element referenced (as defined by that element's ID attribute).
>
> **Entity.** A unit of information storage that contains part of a document (sometimes all of one). An entity can be referenced by a name from one or more places in a document, thereby causing its information to be included in the document at the points of reference. Entities are independent of elements. For example, an entity could contain part of a picture element or one and one-half paragraph elements.*

More concretely, elements mark up characters, attributes qualify how that text is displayed or used, and entities represent specified strings or files. Some attributes, such as ID, are implied, meaning that they may be supplied by an SGML application or an

* ISO/IEC 10744:1992(E), pp. 13–14.

intermediate program, while others may be required. Attributes may also have default values, in which case the attribute need not be supplied explicitly.

DTDs may refer to exterior files that contain general entities, or strings that indicate text or symbols unavailable on the keyboard that are to be supplied by the SGML application (such as `&`, used for the ampersand character). They may also employ convenient groupings of element, called parameter entities. In the DocBook DTD these parameter entites are given names ending in .gp, such as %basechar.gp;. approach simplifies the construction and maintenance of the DTD, but complicates the task of reading it.

Parts of a DTD

A DTD begins with the SGML declaration. It specifies which features of SGML are used, the value of various parameters that are significant to SGML parsers and applications, and the definition of character sets. This section defines the character sets used in the DTD and specifies special SGML features used in the document instance to which the DTD is to apply. The declaration for the DocBook DTD is about 60 lines long, beginning with

```
<!SGML "ISO 8879:1986"
```

and ending with a > on a line by itself. In the DocBook DTD SGML declaration the TOTALCAP, GRPCAP, NAMELEN, TAGLVL, LITLEN, GRPGTCNT, and GRPCNT values have been raised from their defaults in the SGML Reference Concrete Syntax. SHORTTAG has been set to YES, meaning that the end tag of an element may be indicated simply as </>.

The DocBook DTD begins with a group of parameter entities (referencing several ISO standards and a local file of entities). The ISO standards define strings for specifying special characters; these are used if you cannot type the characters on the keyboard or if the characters have a special meaning for SGML, as is the case with the angle brackets and the ampersand.

Next comes a section of parameter entities. Here reusable groups of elements are established for convenience of use within the DTD. Thus, %inlinechar.gp; contains all the elements that may appear in-line in regular text. It is then unnecessary to repeat this list of elements in the element definitions proper: a TableCell, for example, is defined as containing any number of occurrences of %inlinechar.gp;.

Finally comes the section marked "Document Structure," in which the elements used to mark up the text are defined.

DocBook Definitions

The definitive guide to any DTD must be the DTD itself. You don't need to master SGML to get help from a DTD. You can find the allowable *contents* of an element by looking for its definition. You can search for 'NT elementname', so for the element Figure, search for 'NT Figure'. You will find:

```
<!ELEMENT Figure    - - (Title, TitleAbbrev?, ((%figurecontent.gp;)+)*)>
   <!ATTLIST Figure id ID #IMPLIED
      Height CDATA #IMPLIED
      Width CDATA #IMPLIED
      Align (l|c|r) l
      Float (YES|NO) NO >
```

The first line defines what a Figure must and may contain, the second line begins its list of attributes, and the next four lines continue that list. Note that the Height and Width attributes are implied: that is, the SGML application supplies a default value for them if none is provided. The Align and Float attributes have a default (l and NO), which are supplied by the SGML parser if no value is supplied for them.

Then follow out the parameter entities in the definition: look for their definitions by searching for '% entityname'. In this instance you can search for '% figurecontent.gp;', finding:

```
<!ENTITY % figurecontent.gp "(((%object.gp;))|(Oops)|(Comment)|
   (IndexTerm))">
```

and to fill out the definition of Figure you must also look at %object.gp:

```
<!ENTITY % object.gp "((Graphic)|(Equation)|(Table)|(Admonition)|
   (BlockQuote)|((%literallayout.gp;)))"
```

To find all allowable *contexts* of an element, you must look for all occurrences of the element in the DTD; if it occurs within a parameter entity look for all occurrences of that entity, too.

A Structural View of the DTD

This section discusses how the syntactic elements defined in the DocBook DTD may be combined, using examples of text tagged in SGML. The elements defined in the DocBook DTD can be classified in four categories. There are elements that indicate the hierarchy of the information. At the root of this hierarchy is the DocBook element. On the first level of sectioning, there can be one or more book components, such as Chapter or Appendix. Below these sections, there are additional levels of sectioning, marked as Head1 through Head5. These may be nested up to five levels; and each section must begin with a Title.

In addition to its hierarchical organization, the information may be marked up with elements that indicate nonlinear relationships, including IndexTerms, Xrefs (cross references), and Links.

A section is further organized by block-oriented elements. The Para (paragraph) is the most common block-oriented element. Others, such as the various types of lists, are more complex, requiring other elements within them. An item in a list may contain more than one Para, for example, and lists may be nested within lists.

The DTD also recognizes various semantic sets, some of which are peculiar to computer documentation. These semantic set identifiers allow you to label specific terms or usage.

For instance, the word "date" may be classified as a Command, and you may wish to treat all of these elements as a set, whether for formatting or retrieval. Other set identifiers include Function, Quote, and UserInput.

For each element, this document describes what it *must* contain and what it *may* contain—that is, required and optional elements.

Please note:

- Element names in SGML are not case-sensitive.
- In the examples that follow, some punctuation and mathematics characters are represented by General entities (`&`). These entities are defined by external reference in the DTD.
- A few elements appear in various contexts: Title is used for the contents of headings at all levels.
- The typography of the examples is designed for legibility; line breaks and leading white space are not significant.

Book Components and Sections

A book consists of various components, but the DocBook DTD does not require that a valid document be an entire book. Any of the following can be a valid DocBook document on its own: TitlePage, ToC (Table of Contents), LoT (List of Titles), Preface, Chapter, Appendix, Bibliography, Glossary, Index, or Reference. Any set of these elements can also be a valid DocBook document.

The fundamental tag for documents tagged according to the DocBook DTD is the root element, DocBook. This element encloses all other markup in the document (except for documents that have a DOCTYPE declaration at the very outset, indicating the DTD according to which they are marked up.)

```
<DOCBOOK>
    <CHAPTER ID="CH-061-ch05-1">
        <TITLE> An Introduction to the X Window System
        </TITLE>
        <TITLEABBREV>Intro to X
        </TITLEABBREV>
        <PARA>
        And so on.
        .
        .
        .
        </PARA>
    </CHAPTER>
</DOCBOOK>
```

Preface, Chapter, Appendix, Bibliography, and Glossary

A Chapter or Appendix must have a Title and some text—at least a Para or a list—and may have a TitleAbbrev (abbreviated title). Chapter and Appendix have a Number attribute, to be used when the number of the chapter is hard-coded in the input file.

A Preface, Glossary, or Bibliography must have a Title, and has no Number attribute. Thus you may use the Preface element for a foreword or acknowledgements section, if you like, and you may call a Glossary or Bibliography something else, such as “Key Words,” or “Sources.”

Every book component element (Preface, ToC, LoT, Chapter, Appendix, Bibliography, Glossary, Index, and Reference) has an ID attribute.

A Preface, Chapter, Appendix, Bibliography, or Glossary may consist of:

- Highlights (list of main points)
- an Epigraph
- Paras
- Sections (Heads)
- Lists
- Admonitions, Tables, Figures, Equations, BlockQuotes
- Examples, LiteralLayouts, Screens
- Graphics
- Sidebars
- Comments
- Terms to be indexed

Indexes

The Index book component is intended to apply to existing indexes, and may also accomodate indexes prepared from SGML-formatted documents.

An Index has no Title (the title “Index” is to be supplied by the SGML application). It may commence with one or more Paras, and must contain IndexEntries. (An IndexEntry is part of an Index, whereas and IndexTerm is part of the text of a document, marking the place where the term to be indexed occurs.) IndexEntries may have Links. An IndexEntry may include some of the semantic set identifiers found in paragraphs (see “In-Line Components,” below), and may have a Link.

For the use of IndexTerms, see below, *s.v.*

Table of Contents

A ToC must have a Title and may have a TitleAbbrev, followed by ToCEntry1s, which may contain ToCEntry2s. A ToCEntry2 may contain ToCEntry3s, and so on down to ToCEntry5s. ToCEntry1–5s have a PageNum attribute, for use in case you wish to indicate the page numbers on which chapters or sections begin in a printed book. Alternately, a Link may be included as part of the contents of a ToCEntry, both pointing to a section by its ID and containing as text the page number on which the section begins.

```
<DOCBOOK>
    <TOC ID="CH-061-toc-1">
        <TITLE>Table of Contents
        </TITLE>
        <TOCENTRY1>Chapter One: Nuts and Bolts
            <LINK Target="H1-061-ch01-1">
            </LINK>
        </TOCENTRY1>
        <TOCENTRY1>Chapter Two: Screws and Nails
            <TOCENTRY2>Screws
            </TOCENTRY2>
            <TOCENTRY2>Nails
            </TOCENTRY2>
        </TOCENTRY1>
    </TOC>
</DOCBOOK>
```

Lists of Titles (LoT)

An LoT is like a ToC except that it is used for lists of tables, figures, and so on. An LoT contains LoTEntries, which may contain Links, just like ToCEntries. LoTEntries also have a PageNum attribute, but have no subelements like ToCEntry2.

Title Page

A TitlePage may consist of any number of any of these elements, in any order: Author, Date, Copyright, ISBN, LegalNotice, RevHistory, TitleBook, Editor, Edition, PrintHistory, Publisher, and Series.

Sections (Heads)

A Preface, Chapter, Appendix, Glossary, or Bibliography may contain up to five levels of hierarchical sections called Head1, Head2, etc. In a RefEntry, these are RefHead1 and RefHead2 (no further subheads). Paras, lists, and other block-oriented elements may actually be inserted before, between, and after a Head1; in practice, block-oriented elements that are not enclosed within a Head appear only before the first Head1, at the outset of a book component.

A section includes the title of the heading itself and all material following, up to the end of the document or the next section at the same or higher level. Consequently, anything may occur in a section except another book component (TitlePage, Glossary, etc.) or another section of the same level.

The DTD requires that headings be nested properly: it will not allow a Head3 directly under a Head1 without a Head2 in between.

A section must have a Title. Thus,

```
<HEAD1 ID="H1-061-ch05-1">
    <TITLE> Running the Program
    </TITLE>
    <PARA> If you have the sample programs and a workstation that runs X, you can try
    out this program by compiling <FILENAME>basic/basicwin.c</FILENAME>.
    </PARA>
</HEAD1>
```

Sections have a PageNum attribute, for use in case you wish to indicate the page numbers on which sections begin in a printed book. One might process this information into an existing document after it is tagged in SGML; while they are artifacts of its print presentation, page numbers may be useful in an online presentation, too.

Block-Oriented Elements

Most book components can have:

- Highlights
- Epigraphs
- Paragraphs
- Lists
- Tables
- Figures
- Literal layouts
- Examples
- Screens
- Block quotes
- Sidebars
- Footnotes
- Admonitions

Highlights (a section of major points appearing in the book component) and Epigraphs (a section of poetry or prose, usually specially formatted) typically appear at the head of a book component, either before or immediately after the Title.

Paragraphs

Broadly speaking, paragraphs (Paras) are the lowest level of ordinary text that need be tagged except for semantic set identifiers (sentences within paragraphs are not tagged, for example). The DocBook DTD does not distinguish among different types of paragraphs: indented, nonindented, continuation, and hanging.

A continuation paragraph, following a phrase set off from the main text or a block quote, may logically be part of the previous paragraph, but in the DocBook DTD no provision has been made for tagging it distinctively: you must simply tag the two parts of the paragraph as two Paras. See the example below, *s.v.* Example.

In some cases, such as Admonitions and Footnotes, Para tags may be required within the outer tags, to allow such an element to contain multiple paragraphs.

```
<ADMONITION Heading="Warning">
    <PARA> Never use software obtained from sources you would not trust with your
    firstborn.   If in doubt about a program, don't use it at all.
    </PARA>
</ADMONITION>
```

Lists contain ListItems, which typically contain paragraphs; ListItems may also contain more than one paragraph. (See List examples below.)

Aside from straight text, paragraphs may contain any of the following tags for semantic set identifiers (see the "In-line Components" section and for definitions, the alphabetical list of elements, below): VarParam, FootnoteRef, Command, Equation, Filename, Flag, Function, Hardware, Interface, CiteBook, CiteChap, CiteSect, FirstTerm, GlossaryTerm, IndexTerm, ComputerOutput, UserInput, Subscript, Superscript, Quote, Citation, Footnote, Link, XRef, Emphasis, Graphic, OopsChar, WordAsWord, and SystemItem.

Lists

Lists may be of four types:

- VariableList (composed of paragraphs or sets of paragraphs, etc., with associated Terms)
- ItemizedList (bulletted, dashed, etc.)
- OrderedList (numbered or lettered)
- AdmonitionList (a collection of admonitions in list form)

Lists may be nested within ListItems (an ItemizedList inside a ListItem of an OrderedList, for example), and ListItems may include Paras, Equations, BlockQuotes, Tables, Figures, Screens, LiteralLayouts, and Admonitions.

ItemizedLists use an attribute, Mark, to determine the character that precedes each ListItem. You must supply the appropriate character entity (*e.g.* •) as the value of Mark; there is no default.

Likewise, OrderedLists use an attribute, Numeration, to determine how the list is numbered. Numeration may have the value of Arabic (1, 2, 3), Upperalpha (A, B, C), Loweralpha (a, b, c), Upperroman (I, II, III), or Lowerroman (i, ii, iii). The default is Arabic. If your list is numbered beginning with 1, you need not supply a value for Numeration. OrderedList also has an attribute, Continuation, with values Yes or

No, which may be used to indicate whether the numbering of a list begins afresh (No) or continues that of the immediately preceding list (Yes). The default is No, so you need supply the Continuation attribute only if your list continues the numbering of the preceding list.

Normally an OrderedList or ItemizedList nested within a list of the same type is given a different mark (as in outline style). The DocBook DTD does not define any default sequence of marks, so you must either indicate them by using the appropriate attributes or instruct your SGML application that, *e.g.* the marks for an ItemizedList that contains another ItemizedList are to be bullets, then dashes.

An ItemizedList:

<ITEMIZEDLIST MARK="DASH">
<LISTITEM>
<PARA> This function should connect the client to an X server with <FUNCTION>XOpenDisplay</FUNCTION>, and exit gracefully if the connection could not be made.
</PARA>
</LISTITEM>
<LISTITEM>
<PARA> Get information about the physical screen, and use it to calculate the desired size of the window.
</PARA>
<PARA> A second paragraph occurring within the same bulletted ListItem would be marked up as this paragraph is.
</PARA>
</LISTITEM>
</ITEMIZEDLIST>

An OrderedList:

<ORDEREDLIST>
<LISTITEM>
<PARA> The window is mapped with <FUNCTION>XMapWindow</FUNCTION> or related routines.
</PARA>
</LISTITEM>
<LISTITEM>
<PARA> All its ancestors must be mapped. This condition is always satisfied for the children of the root window, the top-level windows of each application.
</PARA>
</LISTITEM>
<LISTITEM>
<PARA> The window must not be obscured by visible sibling windows or their ancestors—this depends on the stacking order. When first mapped, a window appears on top of its siblings, which will be on top of all windows if its parent is the root window.
</PARA>
</LISTITEM>
</ORDEREDLIST>

A VariableList:

<VARIABLELIST>
<TERM>Window
</TERM>
<LISTITEM>
<PARA> A unique integer identifier (ID) that is returned by XCreateWindow or XCreateSimpleWindow and is thereafter used by the program to refer to the created

```
        window resource.
        </PARA>
    </LISTITEM>
    <TERM>Display
    </TERM>
    <LISTITEM>
        <PARA> A large structure that contains information about the server and its
        screens. It is filled only after this program connects to a server by calling XOpenDis-
        play.
        </PARA>
        <PARA> A small structure containing information about the server and its screens
        would be more economical, but we mention it only to generate another paragraph.
        </PARA>
    </LISTITEM>
</VARIABLELIST>
```

In this last example note that the Terms and ListItems are not enclosed by another tag. A ListItem requires Paras or other block-oriented elements nested within it.

Tables

A Table is an ordered array of text, usually with a title and headings. This information may be represented in SGML, but for some uses it may also be convenient to provide it in graphical form. The DocBook DTD thus allows you to mark up a table in SGML and also provide a graphical representation of it in an external file. The definition of Table offered here may be changed to suit SGML-based document formatters now coming on the market.

A Table may have a Title, a TitleAbbrev, and one or more Graphics (which may be pointers to external files that contain the Table formatted in a different way); it may also have a TableSpec (in which typesetting information or hints may be saved) and one or more TableColHeads (a row of headings above each column). It must have TableRows of TableCells. Table has an ID attribute.

Graphic has an attribute, Type, that indicates how the external file is formatted if an external file is indicated by providing an entity as the content of Graphic. Type may have the value GIF, TIFF, DVI, EPS, tbl, Tex, or bitmap. The default is GIF. If the external file is a GIF file you need not supply the Type attribute. Graphic points to an external file if an entity is provided as its content; such entities are to be collected in a table in some other file, where the association of the entities with actual pathnames and filenames may be made more conveniently.

Here is a sample Table:

```
<TABLE ID="TB-061-ch05-1">
    <TITLE> Geometry Specification x and y Offsets
    </TITLE>
    <GRAPHIC TYPE="bitmap"> &061ch05t1;
    </GRAPHIC>
    <TABLESPEC>
    tab (@) linesize(2);
    l — lw(4i).
    </TABLESPEC>
    <TABLECOLHEAD>
        <TABLECELL>Offset Variables
        </TABLECELL>
        <TABLECELL>Description
        </TABLECELL>
    </TABLECOLHEAD>
    <TABLEROW>
        <TABLECELL>+xoff
        </TABLECELL>
        <TABLECELL>A positive x offset specifies the distance by which the left edge of
        the window is offset from the left side of the display.
        </TABLECELL>
    </TABLEROW>
    <TABLEROW>
        <TABLECELL>+yoff
        </TABLECELL>
        <TABLECELL>A positive y offset specifies the distance by which the top edge of
        the window is offset from the top of the display.
        </TABLECELL>
    </TABLEROW>
    .
    .
    .
</TABLE>
```

Figures

A Figure must have a Title, may have a TitleAbbrev, and may contain one or more of the following: Oops, Comment, IndexTerm, Graphic, Equation, Table, Admonition, BlockQuote, Example, LiteralLayout, or Screen.

Figure has an ID attribute and a set of attributes that provide information for presentation. These are Height and Width (not required); Align, which may have the value l, r, or c for left-aligned, right-aligned, or centered (the default is l); and Float, to show whether the Figure should be anchored where it is in the file or may float to another more convenient location. Float may have the value Yes or No; the default is No. Thus, if the Figure is to be left-aligned and not floating, and you do not want to convey its dimensions to the SGML application, you need supply no values for any of these attributes.

```
<FIGURE ID="FG-061-ch05-2" ALIGN="C" FLOAT="NO">
    <TITLE> The xdmlogwin Display
    </TITLE>
    <GRAPHIC> &061ch05t2;
    </GRAPHIC>
</FIGURE>
```

LiteralLayouts, Examples, and Screens

The DocBook DTD provides three elements for material set off from the main text in which white space and line breaks are to be considered significant. These are LiteralLayout, Example, and Screen. These three elements may contain in-line elements (inlinechar.gp) and Comments—no Paras, lists, Figures, Sidebars, or other block-oriented elements.

LiteralLayout is the simplest. It consists only of the set-off lines:

```
<PARA>...for more information about the United Nations and intergalactic space, why
not ask
</PARA>
<LITERALLAYOUT>
Butros Butros Ghali
Secretary General
United Nations
</LITERALLAYOUT>
<PARA>what he thinks about this issue? Then you can propose renaming the Security
Council the Cosmic Security Council.
</PARA>
```

A Example is a program listing or a segment of one. Aside from its content, a Example differs from a LiteralLayout in that it may be titled: it may have a Title and a TitleAbbrev. A titled Example is typically labelled "Example"; an untitled Example is typically a fragment of a program listing. The actual listing is wrapped in a Screen, LiteralLayout, or ProgramListing element. Example has an ID attribute.

A Screen is a representation of all or part of a screen display, which may be textual or graphical. Screens may be composed in two ways. A Screen may simply contain in-line elements, like a LiteralLayout. If not, it must contain a ScreenInfo and a Graphic. The ScreenInfo is a comment about how the image in the Graphic was created (so you can recreate it if need be), and is not meant to be displayed or printed. A Screen has no Title, but has an ID attribute. If you want a Screen to have a Title, enclose it in a Figure element.

Here's a Screen with text content:

```
<PARA> The following text in the <FILENAME>system.mwmrc</FILENAME> file
creates the Root Menu:
</PARA>
<SCREEN>

    # Root Menu Description
    Menu RootMenu
    {
    "Root Menu"                f.title
    "New Window"               f.exec "xterm &"
    "Shuffle Up"               f.circle_up
    "Shuffle Down"             f.circle_down
    "Refresh"                  f.refresh
    no-label                   f.separator
    "Restart..."               f.restart
    }

</SCREEN>
<PARA> The syntax for defining ...
</PARA>
```

Here's a Screen with graphic content:

```
<SCREEN>
    <SCREENINFO> Press the F1 key and type “catfile” while holding down the Alt,
    Page Down, and F12 keys.
    </SCREENINFO>
    <GRAPHIC TYPE="TIFF"> &061ch05g7;
    </GRAPHIC>
</SCREEN>
```

BlockQuotes

BlockQuote is intended for a quotation set off from the main text, rather than occurring in-line (for an in-line quote, use Quote). In a BlockQuote white space is not significant: the lines wrap. A BlockQuote contains Paras.

Sidebars

A Sidebar is a segment of text isolated from the narrative flow of the main text, typically boxed and allowed to float in the page layout. Magazines and newspapers contain many sidebars. A Sidebar must have a Title, may have a TitleAbbrev, and must have text in the form of Paras, Lists, Figures, or Tables; no Heads are allowed. Sidebar has an ID attribute.

Footnotes

Footnotes may contain Paras, lists, and other block-oriented elements. Footnotes may be handled in various ways by the application that reads the SGML-tagged file, and in various documents they may occur in various locations in relation to the footnote mark. The location of the mark should be indicated with FootnoteRef, which is like Link in that it points to the corresponding Footnote. FootnoteRef thus has a Target attribute, the value of which is the ID of the associated Footnote; it also has a Mark attribute, for use in case you wish to indicate the text of the mark itself (typically *). To supply a target for FootnoteRef, Footnote has an ID attribute.

Admonitions

Admonitions are those special sections of text in which a documentation writer tries to call attention to some pitfall in the operation of hardware or software. These sections are typically titled Caution, Note, Warning, Tip, and so on. The text of an Admonition may be composed of Paras, LiteralLayouts, Lists, Tables, or Figures. The heading of the Admonition is supplied through the attribute Heading, for which you must supply a value; there is no default. See the example of a warning, above.

Elements Indicating Nonlinear Relationships

Comments, IndexTerms, Links, Anchors, and XRefs all indicate nonlinear relationships within the text—unlike the hierarchical nesting of sections and the linear flow of paragraphs within sections.

Comments

A Comment is a section of text that is not meant to be displayed or printed, such as a message to a production editor. It is possible to specify that sections of an SGML document be ignored by the SGML application, but Comment should be used for material that is not a part of the text of the document itself.

IndexTerms

IndexTerms are words or phrases to be indexed. It is intended that IndexTerm be used for words or phrases that have been extracted from the text and tagged separately (see above, *s.v.* Indexes and IndexTerms). The DocBook DTD provides for three levels of indexing, as well as for "see" and "see also" indexing. IndexTerms may occur almost anywhere.

IndexTerm has an ID attribute and another optional attribute, Span, that is to be used when it is desired that the index entry for the word or phrase list a span of pages (foobar, 24–37). The value of Span may be Start or End. The attribute PageNum is supplied in case you wish to indicate the page number on which the indexed item appears in a print book.

```
<PARA> This paragraph deals with two subjects that should be listed in the index: how
to rotate your monitor and how to influence the course of sporting events by using your
keyboard.
</PARA>
<INDEXTERM>
    <PRIMARY>rotating your monitor
    </PRIMARY>
</INDEXTERM>
<INDEXTERM>
    <PRIMARY>monitor
    </PRIMARY>
        <SECONDARY>rotation of
        </SECONDARY>
    </PRIMARY>
</INDEXTERM>
<INDEXTERM>
    <PRIMARY>keyboard
    </PRIMARY>
        <SECONDARY>use of
        </SECONDARY>
            <SEEALSO> point shaving
            </SEEALSO>
    </PRIMARY>
</INDEXTERM>
```

IndexEntries might be constructed by processing the items in the text that have been tagged for indexing with IndexTerm and its allowable subelements Primary, Secondary, Tertiary, See, and SeeAlso. In theory, you could tag terms for indexing where they lie in the text, but the DocBook IndexTerm mechanism is intended for separate IndexTerms, in which the term is given as it should appear in the index (which may be with a different capitalization or grammatical form than appears in the text).

Links and Anchors

A Link is a pointer to another section of a document, or another document. It has a Target attribute, which is the ID of the element linked to, and must have some text, which may include semantic set identifiers. It also has a Type attribute, which has been left undefined and is not required. The text of the Link is what is to appear in an on-line display as the text a user clicks on to use the Link:

```
<LINK TARGET="H1-061-ch05-1"> Click here
</LINK>
```

Anchor is a semantic set identifier that marks a location as a target for a Link. Like FootnoteRef, it is an empty element: only the start tag is supplied, with an ID. There is an Anchor <ANCHOR ID="061-ch05-AN-7"> in this sentence.

Cross References (XRefs)

An XRef is a cross reference to another part of a document. It has a Target attribute, which is the ID of the element referred to. An XRef may have content or may be empty. If it is empty only the start tag is supplied, and the text displayed by the XRef will be the content of the element referred to. XRef has a Type attribute that indicates what sort of element is referenced.

It is important to note that while the Target of an XRef will typically be the ID of the Title of an element, the Type attribute names the element itself: Figure, Example, Table, Footnote, Chapter, Section, or Sidebar.

```
See <XREF TARGET="H1-061-ch05-1" TYPE="SECTION">, for more information.
```

This references the text of the heading that has the ID H1-061-ch05-1: "See *Terminal Emulation and the xterm Terminal Type* for more information." The SGML application may be instructed to find the dynamically supplied number of the relevant section (using the information supplied by the Type attribute), and prefix it, appropriately punctuated, to the string supplied by the Target attribute, to produce: "See Section 5.1, *Terminal Emulation and the xterm Terminal Type*, for more information."

Semantic Set Identifiers

Semantic set identifiers are used to label specific terms or usage. Semantic set identifiers may appear anywhere in a block-oriented element, but may not span two block-oriented elements. In the documentation we are converting these elements are typically indicated by font changes.

These are the semantic set identifiers: VarParam, Command, Filename, Flag, Function, Application, Hardware, Interface, CiteBook, CiteChap, CiteSect, FirstTerm, GlossaryTerm, ComputerOutput, UserInput, Subscript, Superscript, Quote, Citation, Emphasis, OopsChar, WordAsWord, and SystemItem. All of these elements are defined in the alphabetized list of elements, below.

Command, Filename, Flag, Function, Subscript, Superscript are to be used for computer commands, names of computer files, options to commands, functions in a subroutine or library, subscripts, and superscripts. SystemItem includes such things as environmental variables, function parameters, macro names, and other system-related items. Hardware and Interface are to be used for names of computer equipment and parts of a GUI, respectively. VarParam tags generic text that the user is to replace with some specific string. UserInput and ComputerOutput are to be used to indicate strings the user types in at the keyboard and text the computer returns (on the display or otherwise). FirstTerm and GlossaryTerm may be used to tag the first occurrence of a word and a word that is glossed in the Glossary, respectively. Quote is to be used for an in-line quotation (otherwise, use BlockQuote). WordAsWord is to be used when a word, letter, or number stands for itself ("the number <WORDASWORD>2</WORDASWORD>"). OopsChar is an escape hatch that may be used for any in-line item in your input that you cannot match in this DTD, *e.g.* during filtering. Emphasis is for words to be emphasized visually: "do <EMPHASIS>not</EMPHASIS> use your computer in the shower."

CiteBook, CiteChap, and CiteSect are for references to other books, chapters, and sections, when the titles of these referred-to items are present in the text:

> <PARA>...You may wish to consult Emmeline Lott's <CITEBOOK>Harem Life in Egypt and Turkey</CITEBOOK> for further information on the Grand Pasha's winter quarters. </PARA>

Citation is for a scientific-style citation in the text:

> <PARA> ...as mentioned in Verdeuzeldonk <CITATION>1979:2</CITATION> and by other authors.

Reference Entries

A man page or reference page is a distinct form of reference information. We have modelled our RefEntry element on man pages. In this DTD, a RefEntry may be part of a Chapter or Appendix, at the same level as a Head1, or a collection of RefEntries may be made into a book component called Reference.

RefEntry Basics

A RefEntry must begin and end with the tags <REFENTRY> and </REFENTRY>.

A RefEntry contains, in order:

- one or more Comments (optional)
- one RefMeta (required)
- one or more Comments (optional)
- one RefNameDiv (required)
- one or more Comments (optional)
- one RefSynopsis (optional)
- one or more RefHead1s (required)

Comments and IndexTerms may also appear anywhere within any division of a RefEntry.

RefEntry Variations

Reference pages have conventional divisions, but no single definitive conventional format. The arrangement of information in the initial divisions of a reference page may be rather complicated. RefEntry accomodates several alternate structures for this information. For example, a typical reference page has only a single subject (here, RefName), which is named in a division that is usually called "Name" (here, RefNameDiv), and is followed by an em dash and a short descriptive phrase (here, RefPurpose). Thus a reference page for *mwm,* the Motif window manager, might have this text in its Name division:

> mwm — the Motif window manager

This is simple to tag: "mwm" is a RefName and the descriptive phrase is a RefPurpose (the em dash need not be represented in the SGML document).

Other reference pages may provide a purpose statement on more than one subject (command, function, file format, and so on). For example, in a reference page for *grep* there may be several subjects described by the same purpose statement:

> grep, egrep, fgrep — search a file for a string or regular expression

Here all the terms in the series are RefNames, and should be tagged individually.

Still other reference pages may provide a group name for a series of subjects that is not itself one of the subjects discussed. For example, in an AT&T reference page called “string,” the name of the reference page is given in this manner:

string: strcasecmp, strncasecmp — string operators

In this series, “string,” is not a function name like the terms that follow the colon following “string.” In a RefEntry the word “string” is tagged as a RefDescriptor, which is an optional rather than a required element of RefNameDiv.

Many of the higher-level elements of RefEntry may occur only within RefEntries, not within Chapters or other book components. These are: RefMeta, RefFileName, RefSection, RefMiscInfo, RefNameDiv, RefDescriptor, RefName, RefPurpose, RefSynopsis, RefSynopsisblock, RefHead1, and RefHead2.

RefMeta

The RefMeta must contain, in order, a RefFileName and a RefSection, followed by one or more RefMiscInfos.

All this is metainformation, some of which may occur in the heading of a reference page. None of it is from the body of a reference page. RefFileName is the primary name by which the reference page is sorted and indexed. RefSection is the alphanumerical designation often supplied after the RefFileName in parentheses, as in “string(3C),” and refers to the historical arrangement of reference pages in sections. Various other pieces of information may occur following the RefSection; each of them may be captured with the tag RefMiscInfo, so that you may have a series of RefMiscInfos (as many as eight have been reported in legacy documents). For example, O’Reilly’s reference page for XmuReshapeWidget (vol. 5), includes the heading “Xmu—Shape Extension Utilities,” which should be tagged as a RefMiscInfo. All of these RefMeta elements may contain only text, without further nested elements. Examples are copyright, date, draft status, hardware architecture, and a descriptive phrase for use in a print header.

Do not allow yourself to become confused by the fact that RefFileName may contain the same contents as RefName, in the RefNameDiv (for the *mwm* reference page, “mwm” is the content of both RefFileName and RefName. This may not always be the case, as in the “string” example above: in that instance, “string” is the content of RefFileName, but it is not a RefName. Remember, the RefMeta does not contain the contents of the Name division of a reference page, only metainformation.

```
<REFMETA>
    <REFFILENAME> ldsysdump
    </REFFILENAME>
    <REFSECTION> ldsysdump(1M)
    </REFSECTION>
    <REFMISCINFO> System Administration Commands
    </REFMISCINFO>
    <REFMISCINFO> Copyright 1985
    </REFMISCINFO>
</REFMETA>
```

RefNameDiv

RefNameDiv is the top division of a reference page, the one usually labelled “Name.”

The RefNameDiv contains, in order, an optional RefDescriptor (for reference pages such as the "string" example, above), one or more RefNames, and a RefPurpose. RefDescriptor and RefName may contain further nested elements from the parameter entity group refname.content.gp, including the contents of basechar.gp: plain text, VarParam, Command, Filename, Function, Hardware, Interface, and OopsChar.

RefPurpose may contain anything in inlinechar.gp, including the contents of basechar.gp: plain text, VarParam, Command, Filename, Flag, Function, Interface, Interface, Hardware, ComputerOutput, UserInput, Subscript, Superscript, CiteBook, CiteSect, Quote, IndexTerm, Link, FirstTerm, GlossaryTerm, XRef, OopsChar, and Emphasis.

The examples of RefNameDiv given above would be tagged:

```
<REFNAMEDIV ID="061-RP72">
    <REFNAME> mwm
    </REFNAME>
    <REFPURPOSE> the Motif window manager
    </REFPURPOSE>
</REFNAMEDIV>

<REFNAMEDIV ID="061-RP73">
    <REFNAME> grep
    </REFNAME>
    <REFNAME> egrep
    </REFNAME>
    <REFNAME> fgrep
    </REFNAME>
    <REFPURPOSE> search a file for a string or regular expression
    </REFPURPOSE>
</REFNAMEDIV>

<REFNAMEDIV ID="061-RP74">
    <REFDESCRIPTOR> string
    </REFDESCRIPTOR>
    <REFNAME> strcat
    </REFNAME>
    <REFNAME> strdup
    </REFNAME>
    <REFNAME> strncat
    </REFNAME>
    <REFNAME> strcmp
    </REFNAME>
    <REFNAME> strncmp
    </REFNAME>
    <REFPURPOSE> string operators
    </REFPURPOSE>
</REFNAMEDIV>
```

RefSynopsis

RefSynopsis is intended for the second division of reference pages, usually titled Synopsis or Syntax. But there are reference pages without Synopsis or Syntax divisions, so RefSynopsis is an optional part of RefEntry. The content of this division is often simply a single line giving the syntax for the subject of the reference page, but it may also be a set of lines in which white space and line breaks are significant. In some cases subheadings occur in this division. Thus the content of RefSynopsis may be a RefSynopsisblock (containing a single line or a set of lines), or a RefSynopsisblock followed by any number of RefHead2s, or one or more RefHead2s.

Both RefNameDiv and RefSynopsis have an ID attribute.

Here is a valid RefSynopsis:

```
<REFSYNOPSIS ID="RS-061-66">
    <REFSYNOPSISBLOCK>
        <COMMAND> ls </COMMAND>
        [<VARPARAM> options </VARPARAM>] <FILENAME>
        <VARPARAM>directory_name</VARPARAM><FILENAME>
    </REFSYNOPSISBLOCK>
</REFSYNOPSIS>
```

RefHeads

The rest of a reference page consists of one or more divisions, which are to be tagged as RefHead1s, which may have RefHead2s. These are just like the Head1 and Head2 of the DocBook DTD; with the same required and allowable contents.

Miscellaneous

Some elements are not part of the illustrations or body of text in a printed work. Such elements may appear in a printed work but are not part of the text. Others are provided in case they should be needed for some aspect of on-screen display or book production (TitleAbbrevs). Still others are metainformation, such as comments.

TitleAbbrev may be provided where a long Title might be inappropriately truncated in some on-line display such as a title bar, or for use in a printed header or footer.

RevHistory, an element of TitlePage, records the authorship of changes to a book. It consists of one or more Revisions, each of which must contain a RevNumber, a Date, one or more AuthorInitials, and a RevRemark.

The Elements Alphabetized

- Admonition is a Warning, Caution, Note, Tip, or Hint. Its heading is supplied by its Heading attribute, the value of which you must supply; there is no default. The contents of an Admonition may include Paras, lists, and other block-oriented elements.
- AdmonitionList is a collection of admonitions in list form. Its contents are ListItems, not Admonitions (though ListItems may contain Admonitions).
- Appendix may contain anything found in a Chapter. It has an ID attribute.
- Author is part of TitlePage and consists of one or more optional Firstnames and an optional Surname element.

```
<AUTHOR><FIRSTNAME>David Midway<FIRSTNAME>
<SURNAME>Flanagan</SURNAME></AUTHOR>
```

- Anchor is an empty tag, containing only an ID attribute, that marks a target for a Link. It may appear almost anywhere.
- AuthorInitials may occur in Revisions: these indicate the author of the revision, not the book.
- Bibliography is a book component. It must have a Title, may have a TitleAbbrev,

and may contain Heads, in case the bibliography is divided into sections. It has an ID attribute.

- BlockQuote is a quotation set off from the main text, rather than occurring in-line. It may be titled.
- BookAcronym is an optional part of TitlePage, and may be used alongside or in preference to TitleAbbrev.
- Chapter may contain anything except elements restricted to TitlePage, ToC, Glossary, Bibliography, Index, or RefEntry. It has an ID attribute.
- Citation is an in-line bibliographic reference in scientific style to another published work.
- CiteBook is the name of another book (or journal) cited in the text.
- CiteChap is a reference to a specific chapter of any book cited in the text, including the book in which the CiteChap appears.
- CiteSect is a reference to a specific section of another document cited in the text, including the document in which the CiteSect appears.
- Command is the name of an executable program, or the entry a user makes to execute a command. Command may also be used for a wide range of terms. *Cf.* ComputerOutput. Command has a RefEntry attribute, which states whether the Command has an associated RefEntry (reference or man page); the default is No.
- Comment is remark made within the document file that is intended for use during interim stages of production.
- ComputerOutput is data returned to the user by the computer; *cf.* Systemitem. Generally, ComputerOutput will occur within Screens, but may occur in-line.
- Copyright is part of TitlePage and consists of one or more Years and zero or more Holders.
- Date appears in Revisions.
- Editor is like Author, an element of TitlePage.
- Emphasis is provided for use where one would traditionally employ italics (more rarely, bold) to emphasize a word or phrase.
- Epigraph is to be used for a brief section of poetry or prose at the start of a chapter. It contains Paras.
- Equation is a mathematical equation set off on a line by itself, or occurring in-line. At present it is contemplated that Equations contain Graphics, the contents of which may point to external files that contain the actual equations. Equations may be titled; they have an ID attribute.
- Example is intended for sections of program source code that are provided as examples in the text. Example has an ID attribute. A Example has an optional Title,

optional TitleAbbrev, and a Screen, LiteralLayout, or ProgramListing, in which the actual listing is supplied.

- Figure is an illustration. A Figure may may contain one or more of the following: Graphic, Equation, Table, Admonition, BlockQuote, Screen, Example, and LiteralLayout. A Figure must have a Title, and may have a TitleAbbrev. It has an ID attribute and also has attributes Height, Width, Align, and Float. See the discussion of Figure in the first part of this document.
- Filename is the name of a file, including pathname if this information is present. Filename has a RefEntry attribute, which states whether the Filename has an associated RefEntry (reference or man page); the default is No.
- Firstname is part of Author and Editor. See Author.
- FirstTerm is to be used for the first occurrence of a word, when the style of the document is such that these first occurrences are differentiated in some manner.
- Flag is a option to a Command (in the Command "ls -l" "-l" is a flag), but is not nested with the Command tags.
- Footnote delimits the text of a footnote. The point in the text where the mark for a specific footnote goes is indicated by FootnoteRef; the text of the Footnote may appear in another location in the file. A Footnote may contain all of the permissible contents of Paras, but ordinarily one would not put Figures or Admonitions in Footnotes. Footnote has an ID attribute.
- FootnoteRef indentifies the location for a footnote mark. It and has a Target attribute the value of which is the ID of the associated Footnote, and a Mark attribute, indicating the contents of the Mark (such as *).
- Function is a subroutine in a program or external library. Function has a RefEntry attribute, which states whether the Function has an associated RefEntry (reference or man page); the default is No.
- Glossary is a book component, which may contain the same things as a Chapter. It has an ID attribute.
- GlossaryEntry is an entry in a Glossary, but is not further defined. Please note that GlossaryEntry occurs in a Glossary, not in the text of (*e.g.*) a Chapter.
- GlossaryTerm is an in-line element for tagging terms in the text of a Chapter (for example) that are glossed in the Glossary. As you may not want to tag all occurrences of these words, you might consider GlossaryTerm to be similar to FirstTerm. Please note that GlossaryTerm occurs outside the Glossary, not within it.
- Graphic may occur within a Figure, Table, Screen, or Equation, to enclose graphical data or to point to an external file containing the contents of one of these elements. Graphic has a Type attribute, the value of which may be GIF, TIFF, DVI, EPS,

tbl, Tex, or bitmap. The default is GIF. If it is desired to point to an external file, an entity must be supplied as the content of Graphic.

- Hardware is any name for a physical part of a computer system.
- Head1 marks a section of a document that begins with a first-level heading. Anything may occur within a Head1 except a book component (TitlePage, Preface, Chapter, Appendix, Glossary, Bibliography, Index, or Reference), or another Head1. Head1 must have a Title, which is the text of the heading itself, it may have a TitleAbbrev, and must be include some block-oriented elements. Head1–5 have ID attributes and PageNum attributes, for indicating the page on which a Head begins in print.
- Head2 is a section beginning with a second-level heading, and must be nested within a Head1. Anything may occur within a Head2 except a Head1, another Head2, or a book component. HeadIDs, Titles, and required content for Head2s are just like those for Head1s.
- Head3. *Cf.* Head2.
- Head4. *Cf.* Head2.
- Head5. *Cf.* Head2. No further subdivisions are supplied in the DocBook DTD.
- Highlights is a list of main points at the beginning of a book component such as a Chapter. It may contain Paras and lists, and has an ID attribute.
- Holder is part of Copyright and is the name of the holder of the copyright of the document.
- Index is a book component, containing Indexentries (not the same as IndexTerms). You could use IndexTerms to create an Index. Index has an ID attribute, the value of which may be a *list* of the IDs of the IndexTerms in the text.
- IndexEntry is a part of Index, containing characters and in-line elements (such as Command or Flag), and Links.
- IndexTerm is a word or phrase to be indexed. Primary, Secondary, and Tertiary index items are nested under this tag, as are See and SeeAlso items. IndexTerm has an ID attribute. For index entries that span a section of text, there is an optional Span attribute, which may have the value Start or End. IndexTerm also has a PageNum attribute, for indicating the page on which the indexed term is found in print.
- Interface is the name of any part of a graphical user interface, such as a button, icon, menu, menu selection, or window.
- ISBN is part of TitlePage, and should be used for the International Standard Book Number of the document if it has one. (You could use it for an ISSN, also.)
- ItemizedList is a list in which each item is marked with a bullet, dash, or other dingbat (or no mark at all). It consists of one or more ListItems. A ListItem in an ItemizedList may contain other lists, may be nested within other lists, and may

contain Figures and Tables. ItemizedList has a Mark attribute for which you must supply a character entity (*e.g.* •); there is no the default.

- LegalNotice is part of TitlePage, for acknowledgement of trademarks and so forth.
- Link is a hypertext link. It has text as content, and has a mandatory Target attribute, in which the ID of the object linked to should be provided. It has an optional Type attribute, the use of which is undefined.
- ListItem is a wrapper for the elements of item in an ItemizedList, OrderedList, VariableList, or AdmonitionList. It may contain one or more Paras. In a VariableList, ListItem occurs, with Term, within Termdefinition.
- LoT is the generic tag for a book component such as a List of Figures or List of Tables. It must have a Title, may have a TitleAbbrev, and contains multiple LoTEntries. LoT has an ID attribute.
- LoTEntry is an element of LoT, and contains the text of the Title to be listed (including, if desired, in-line elements such as Command or Flag). LoTEntry has a PageNum attribute, in which you may indicate the numbers of the pages in the hard copy on which the specific entries occur.
- LiteralLayout is the wrapper for lines set off from the main text that are not tagged as Screens or Examples.
- Oops is a catch-all category that can be used to tag any element above the in-line level that is not provided for elsewhere in this DTD. Best used in filtering.
- OopsChar is an in-line Oops.
- OrderedList is a numbered or lettered list, consisting of ListItems. The ListItems in an OrderedList may contain other lists. OrderedList has a Numeration attribute, which may have the value Arabic, Upperalpha, Loweralpha, Upperroman, or Lowerroman. The default is Arabic. OrderedList also has an attribute, Continuation, with values Yes or No, which may be used to indicate whether the numbering of a list begins afresh (No) or continues that of the immediately preceding list (Yes). The default is No, so you need supply the Continuation attribute only if your list continues the numbering of the preceding list. See the example of an OrderedList in the first part of this document.
- Para is a paragraph. See the first part of this document for a discussion of allowable contents. Para has an Indentation attribute, the value of which may be Indent or

Noindent; the default is Indent.

- Preface is a book component, just like a Chapter, *q.v.* Use the Title element to provide the word "Preface" or other string as desired. Preface has an ID attribute.
- Primary is a word or phrase that is to appear in the index under as a primary entry. It must be nested within IndexTerm tags.
- PrintHistory is part of TitlePage, *q.v.* It contains Paras.
- ProgramListing occurs in Example, and contains the actual listing of the program. White space and line breaks are significant in a ProgramListing.
- Publisher is an element of TitlePage, *q.v.*
- Quote is supplied for in-line quotations. For block quotes use BlockQuote.
- RefEntry is the outer wrapper for a reference page. RefEntry has an ID attribute.
- Reference is a collection of RefEntries, with a Title and an optional TitleAbbrev. Reference has an ID attribute.
- RefFileName is the primary name given to the reference page for sorting and indexing. It may be the same as the first of the RefNames, or it may be the same as the RefDescriptor.
- RefHead1 is equivalent to a Head1 in the DocBook DTD. RefHead1 has an ID attribute.
- RefHead2 is equivalent to a Head2 in the DocBook DTD. It has an ID attribute.
- RefMeta is the first major division of a reference page, in which metainformation about the reference page is supplied. RefMeta has an ID attribute.
- RefMiscInfo marks information in RefMeta that may be supplied by vendors, such as copyright, release date, revision date, print status, operating system, hardware architecture, or a descriptive phrase for use in a print header.
- RefName is the name of the subject or subjects of a reference page, and appears within RefNameDiv.
- RefNameDiv is the second major division of a reference page, in which the name (RefName) or names of the subject or subjects of the reference page are given, along with a RefPurpose. RefNameDiv has an ID attribute.
- RefPurpose is a short phrase describing the subject of the reference page. It follows the RefNames in the RefNameDiv division.
- RefSection designates the section of a complete set of reference pages that a reference page belongs to. It appears within RefMeta.
- RefSynopsis is the third major division of a reference page, in which the syntax of the subject of the reference page is indicated. It contains a RefSynopsisblock.

RefSynopsis has an ID attribute.

- RefSynopsisblock marks the line or lines that provide the syntax of the subject of the reference page. It appears within RefSynopsis.
- ReleaseInfo is an element of TitlePage, in which information about a particular version of a document may occur.
- RevHistory is a section of TitlePage consisting of Revisions.
- Revision is an entry in RevHistory, recording some revision made to the text. A Revision has a mandatory Revisionnumber and Date, one or more sets of AuthorInitials, and a Revisionremark.
- Revisionnumber is an element of Revision, *q.v.*
- Revisionremark is an element of Revision, recording what the Revision was.
- Screen is intended for sections of text that are taken from on-line displays. A Screen may consist of text (in which the white space and line breaks are considered significant) or a ScreenInfo and a Graphic. Screen has an ID attribute.
- ScreenInfo is a part of Screen, *q.v.* A ScreenInfo may indicate how the Graphic with which it is paired was created as a guide for future revisions.
- Secondary is a word or phrase that is to appear in the index under a Primary entry. It must be nested within IndexTerm tags and must follow a Primary element.
- See is a word or phrase that is to appear in the index preceded by the word "see" and another word or phrase; it must be nested within IndexTerm tags and must follow an Primary or Secondary element.
- SeeAlso is a word or phrase that is to appear in the index followed by the words "see also"; it must be nested within IndexTerm tags and must follow an Primary or Secondary element.
- Series is part of TitlePage and names the publication series of which the book is a part.
- Sidebar is a segment of text isolated from the narrative flow of the main text, typically boxed. Magazines and newspapers contain many sidebars. A Sidebar must have a Title, may have a TitleAbbrev, and must have text formatted in some fashion: allowable contents are Paras, lists, Figures, and Tables. No Heads are allowed. Sidebar has an ID attribute.
- Subscript tags subscripts.
- Superscript tags superscripts. Two cubed is 2<SUPERSCRIPT>3</SUPERSCRIPT>.
- Surname is an element of Author in TitlePage, supplied so that a group of documents may be indexed by their authors's surnames. Surinam is a country in South America.
- SystemItem includes such things as environmental variables, function parameters, macro names, and other system-related items.
- Table is an array of text. See the first section of this document for further discussion. A Table may have a Title, TitleAbbrev, and Graphic; it may have a

TableSpec (containing such things as troff formatting information), one or more TableColHeads (containing headings for the columns of the Table), and must have TableRows of TableCells. Table has an ID attribute and a Type attribute, the value of which may be Simple, Spannedcells, or Unruled. The default is Simple.

- TableCell is a part of a TableRow in a Table.
- TableColHead is an optional part of Table, containing headings for the columns of the Table as TableCells.
- TableSpec tags formatting information for a Table; it is not the text of a Table's heading.
- TableRow is part of Table. A Table has TableRows of TableCells.
- Term is the hanging term attached to a ListItem within a VariableList; visually, a VariableList is a set of Terms with attached items such as paragraphs. Each ListItem may be associated with a set of Terms.
- Tertiary is a word or phrase that is to appear in the index under a Secondary entry. It must be nested within IndexTerm tags and must follow a Secondary element.
- Title is a string of characters, the text of a heading or element title. It is NOT the title of a book cited in the text or Bibliography. A Title may have a PageNum and must have some text, including in-line tags. Title has an ID attribute.
- TitleAbbrev is an optional element of anything titled. You may want to employ this element when a title is so long that you fear it will be truncated in some element of an on-line display, such as a title bar. It should be a string of characters.
- TitleBook is part of TitlePage, and is the full title of the book as published, including subtitle. Separate the subtitle from the title with a colon.
- TitlePage is a book component, containing one or more of the following: Author, Date, Copyright, ISBN, LegalNotice, RevHistory, TitleBook, Editor, Edition, PrintHistory, Publisher, and Series. No order is enforced among these elements, but you are urged to begin with TitleBook and Author(s). TitlePage has an ID attribute.
- ToC is a Table of Contents, a separate book component. It must have a Title, may have a TitleAbbrev, and contains ToCEntry1s. ToC has an ID attribute.
- ToCEntry1 is the top-level tag for items in a ToC. In the manner of Head1, it may contain subsidiary groups (*e.g.*, for headings and subheadings within a chapter), named ToCEntry2–5. ToCEntry1–5s have a PageNum attribute, for indicating the page numbers on which the ToCEntries appear in a print book.
- UserInput is an in-line element use to tag data entered by the user. You might use this element most frequently in LiteralLayouts and Screens.
- VariableList is a list of Terms with associated items, usually paragraphs. It is possible to bunch together more than one term for a given item. Inclusions are as for OrderedList, *q.v.* See examples in the first section of this document.
- VarParam tags any character string that is to be replaced by a real name or value by the user. In "cat filename," filename is a VarParam. If a VarParam is also, say,

a Command, nest the VarParam tags within the Command tags: VarParam may contain only text, not tags.

- WordAsWord is supplied for use when a word (or letter or number) is used not to represent the thing or idea it usually represents, but merely as the word itself. For example, "The term <WORDASWORD>Gothic</WORDASWORD> means different things to art historians and typographers," or for a single character, "the letter <WORDASWORD>X</WORDASWORD>".
- XRef is a link to another part of the document, such as a Table, Figure, or Example. An XRef must either contain text or have a Target attribute, the value of which is the ID of the referenced element.
- Year is part of TitlePage (year of publication of the book).

Notes on Modifying the DTD

The basic scaffolding of the Docbook DTD contains many entities that are used by its elements to define the contents. These entities can be thought of as names of a set of parentheses. Each of these can then be expanded to change locally. For example, admoncontent.gp defines the contents of an Admonition. If one wanted to include another element to the possible alternatives, such as HelpCard, the proper procedure would be to change admoncontent.gp from:

```
<!-- Contents for the element Admonition -->
<!ENTITY % admoncontent.gp "((Para)|((%list.gp;))|((%object.gp;))|(Oops)|(Comment)|
(IndexTerm))" >
```

to:

```
<!-- Contents for the element Admonition -->
<!ENTITY % admoncontent.gp "((Para)|((%list.gp;))|((%object.gp;))|(Oops)|(Comment)|
(IndexTerm)|(HelpCard))" >
```

To provide another component in the basic scaffolding of the document structure, a developer has to decide where to try to insert the new component.

- Decide where in the basic hierarchy this component occurs: is it as a book component, a heading within a book component, a block within a section, or a semantic set element that occurs within a lot of elements?
- Determine whether, at that level in the scaffolding, the order in which that component occurs is important.
- Insert the new element where a surrounding element can get the new element as content. This may occur in the parent's element definition or in an entity definition that the parent element uses.
- Now the new element needs to defined, in terms of what currently exists (entities and other elements) and what also needs to be added.

It is always a good idea to validate the DTD, looking for ambiguous productions, mixed content and other gotchas…

An example:

I want to add a semantic member called HODGEPODGE so that most elements would contain it within the data stream that is comprised primarily of text. So my level is within paragraph blocks, and even with title information. That basically leaves me with the entity inlinechar.gp or basechar.gp. I don't want HODGEPODGE included in most other members of inlinechar.gp, but as a sibling of them. So I'll choose inlinechar.gp and add it in there with `|(hodgepodge)` at the end of the entity declaration. I'll add an element definition for HODGEPODGE that looks a lot like SystemItem, and I'm done, except to make sure that I didn't syntactically mess up the DTD. Below are what those SGML statements would look like:

```
<!ENTITY % inlinechar.gp "(((%basechar.gp;))|(FootnoteRef)|(Command)|(Filename)|
(Flag)|(Function)|(Application)|(Hardware)|(Interface)|(CiteBook)|(CiteChap)|
(CiteSect)|(FirstTerm)|(GlossaryTerm)|(IndexTerm)|(ComputerOutput)|(UserInput)
|(Subscript)|(Superscript)|(Quote)|(Citation)|(Footnote)|(Link)|(XRef)|
(Emphasis)|(OopsChar)|(wordasword)|(systemitem)|(Hodgepodge)">
...
<!ELEMENT Hodgepodge - - ((%basechar.gp;)+)>
```

```
<!SGML "ISO 8879:1986"
-- SGML declaration with Reference Concrete Syntax modifications --
-- DocBook DTD
Release 1.0, Copyright 1992 HaL Computer Systems, Inc., and
O'Reilly & Associates, Inc.
Permission to use, copy, modify and distribute the DocBook DTD and
its accompanying documentation for any purpose and without fee is
hereby granted, provided that this copyright notice appears in all
copies.  HaL Computer Systems and O'Reilly & Associates make no
representation about the suitability of the DTD for any purpose.
It is provided "as is"without expressed or implied warranty.
The DocBook DTD is maintained by O'Reilly & Associates.  Please
direct all questions, bug reports, or suggestions for changes to
dtd@ora.com or by mail to:  Terry Allen, O'Reilly & Associates, Inc.,
A Morris Street, Sebastopol, California, 95472.
--
CHARSET
          BASESET "ISO 646:1983//CHARSET
             International Reference Version (IRV)//ESC 2/5 4/0"
          DESCSET 0 9 UNUSED
             9 2 9
             2 UNUSED
             1 13
             18 UNUSED
             95 32
             1 UNUSED
CAPACITY SGMLREF
          TOTALCAP 150000
          GRPCAP 150000
SCOPE DOCUMENT
SYNTAX
          SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
             19 20 21 22 23 24 25 26 27 28 29 30 31 127 255
          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 ""
             UCNMSTRT ""
             LCNMCHAR ".-"
             UCNMCHAR ".-"
             NAMECASE GENERAL YES
                ENTITY NO
             DELIM GENERAL SGMLREF
                SHORTREF SGMLREF
             NAMES SGMLREF
          QUANTITY SGMLREF
             NAMELEN 34
             TAGLVL 100
             LITLEN 1024
             GRPGTCNT 150
             GRPCNT 64
```

```
FEATURES
          MINIMIZE
             DATATAG NO
             OMITTAG NO
             RANK NO
             SHORTTAG YES
          LINK
             SIMPLE NO
             IMPLICIT NO
             EXPLICIT NO
          OTHER
             CONCUR NO
             SUBDOC NO
             FORMAL NO
          APPINFO NONE
>
<!DOCTYPE Docbook [
<!-- Publicly declared External Entities -->
<!-- Uncomment the ENTITY declaration for your application.  -->
<!--
<!ENTITY % ISOamsa PUBLIC "ISO 8879:1986//ENTITIES Added Math Symbols:  Arrow
Relations//EN">
%ISOamsa;
-->
<!--
<!ENTITY % ISOamsa SYSTEM "iso-amsa.gml">
%ISOamsa;
-->
<!--
<!ENTITY % ISOamsb PUBLIC "ISO 8879:1986//ENTITIES Added Math Symbols:  Binary
Operators//EN">
%ISOamsb;
-->
<!--
<!ENTITY % ISOamsb SYSTEM "iso-amsb.gml">
%ISOamsb;
-->
<!--
<!ENTITY % ISOamsc PUBLIC "ISO 8879:1986//ENTITIES Added Math Symbols:  Delimiters//EN">
%ISOamsc;
-->
<!--
<!ENTITY % ISOamsc SYSTEM "iso-amsc.gml">
%ISOamsc;
-->
<!--
<!ENTITY % ISOamsn PUBLIC "ISO 8879:1986//ENTITIES Added Math Symbols:  Negated
Relations//EN">
%ISOamsn;
-->
<!--
<!ENTITY % ISOamsn SYSTEM "iso-amsn.gml">
%ISOamsn;
-->
<!--
<!ENTITY % ISOamso PUBLIC "ISO 8879:1986//ENTITIES Added Math Symbols:  Ordinary//EN">
%ISOamso;
```

```
-->
<!--
<!ENTITY % ISOamso SYSTEM "iso-amso.gml">
%ISOamso;
-->
<!--
<!ENTITY % ISOamsr PUBLIC "ISO 8879:1986//ENTITIES Added Math Symbols:  Relations//EN">
%ISOamsr;
-->
<!--
<!ENTITY % ISOamsr SYSTEM "iso-amsr.gml">
%ISOamsr;
-->
<!--
<!ENTITY % ISObox PUBLIC "ISO 8879:1986//ENTITIES Box and Line Drawing//EN">
%ISObox;
-->
<!--
<!ENTITY % ISObox SYSTEM "iso-box.gml">
%ISObox;
-->
<!--
<!ENTITY % ISOcyr1 PUBLIC "ISO 8879:1986//ENTITIES Russian Cyrillic//EN">
%ISOcyr1;
-->
<!--
<!ENTITY % ISOcyr1 SYSTEM "iso-cyr1.gml">
%ISOcyr1;
-->
<!--
<!ENTITY % ISOcyr2 PUBLIC "ISO 8879:1986//ENTITIES Non-Russian Cyrillic//EN">
%ISOcyr2;
-->
<!--
<!ENTITY % ISOcyr2 SYSTEM "iso-cyr2.gml">
%ISOcyr2;
-->
<!--
<!ENTITY % ISOdia PUBLIC "ISO 8879:1986//ENTITIES Diacritical Marks//EN">
%ISOdia;
-->
<!--
<!ENTITY % ISOdia SYSTEM "iso-dia.gml">
%ISOdia;
-->
<!--
<!ENTITY % ISOgrk1 PUBLIC "ISO 8879:1986//ENTITIES Greek Letters//EN">
%ISOgrk1;
-->
<!--
<!ENTITY % ISOgrk1 SYSTEM "iso-grk1.gml">
%ISOgrk1;
-->
<!--
<!ENTITY % ISOgrk2 PUBLIC "ISO 8879:1986//ENTITIES Monotoniko Greek//EN">
%ISOgrk2;
-->
<!--
<!ENTITY % ISOgrk2 SYSTEM "iso-grk2.gml">
```

```
%ISOgrk2;
-->
<!--
<!ENTITY % ISOgrk3 PUBLIC "ISO 8879:1986//ENTITIES Greek Symbols//EN">
%ISOgrk3;
-->
<!--
<!ENTITY % ISOgrk3 SYSTEM "iso-grk3.gml">
%ISOgrk3;
-->
<!--
<!ENTITY % ISOlat1 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN">
%ISOlat1;
-->
<!--
<!ENTITY % ISOlat1 SYSTEM "iso-lat1.gml">
%ISOlat1;
-->
<!--
<!ENTITY % ISOlat2 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 2//EN">
%ISOlat2;
-->
<!--
<!ENTITY % ISOlat2 SYSTEM "iso-lat2.gml">
%ISOlat2;
-->
<!--
<!ENTITY % ISOnum PUBLIC "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN">
%ISOnum;
-->
<!--
<!ENTITY % ISOnum SYSTEM "iso-num.gml">
%ISOnum;
-->
<!--
<!ENTITY % ISOpub PUBLIC "ISO 8879:1986//ENTITIES Publishing//EN">
%ISOpub;
-->
<!--
<!ENTITY % ISOpub SYSTEM "iso-pub.gml">
%ISOpub;
-->
<!--
<!ENTITY % ISOtech PUBLIC "ISO 8879:1986//ENTITIES General Technical//EN">
%ISOtech;
-->
<!--
<!ENTITY % ISOtech SYSTEM "iso-tech.gml">
%ISOtech;
-->
<!-- Parameter Entities -->
<!-- These are reuseable groups of entities and elements, collected to
make
revision, augmentation, and maintenance of this DTD easier.  -->
<!-- Book Components -->
<!ENTITY % bookcomponent.gp "((Preface)|(TitlePage)|(ToC)|(LoT)|(Chapter)|(Appendix)
|(Bibliography)|(Glossary)|(Index)|(Reference))">
<!-- Contents that can appear almost anywhere -->
<!ENTITY % basechar.gp "((#PCDATA)|(VarParam)|(Anchor))">
```

```
<!-- Contents that can appear in most contexts, in-line -->
<!ENTITY % inlinechar.gp "(((%basechar.gp;))|(FootnoteRef)|(Command)|(Filename)
|(Flag)|(Function)|(Application)|(Hardware)|(Interface)|(CiteBook)|(CiteChap)
|(CiteSect)|(FirstTerm)|(GlossaryTerm)|(IndexTerm)|(ComputerOutput)|(UserInput)
|(Subscript)|(Superscript)|(Quote)|(Citation)|(Footnote)|(Link)|(XRef)|(Emphasis)
|(OopsChar)|(wordasword)|(systemitem))">
<!-- Contents for the element VarParam -->
<!ENTITY % varparam.content.gp "((#PCDATA)|(Superscript)|(Subscript))">
<!-- Contents for citations of Titles -->
<!ENTITY % cite.title.content.gp "((#PCDATA)|(Command)|(Filename)|(Flag)|(Function)
|(Application)|(Hardware)|(Interface)|(UserInput)|(ComputerOutput)|(Footnote)
|(Subscript)|(Superscript))">
<!-- Contents for the element GlossaryTerm -->
<!ENTITY % glossterm.content.gp "((#PCDATA)|(Command)|(Filename)|(Function)|(Application)
|(Hardware)|(Interface))">
<!-- Contents for the element UserInput -->
<!ENTITY % uicontent.gp "(((%basechar.gp;))|(Command)|(Filename)|(Flag)|(Function)
|(Application)|(Hardware)|(Interface)|(OopsChar))">
<!-- Contents for the elements Subscript and Superscript -->
<!ENTITY % script.content.gp "((#PCDATA)|(VarParam))">
<!-- Contents for the element Quote -->
<!ENTITY % quote.content.gp "(((%basechar.gp;))|(FootnoteRef)|(Command)|(Filename)
|(Flag)|(Function)|(Application)|(Hardware)|(Interface)|(CiteBook)|(CiteChap)
|(CiteSect)|(GlossaryTerm)|(IndexTerm)|(UserInput)|(ComputerOutput)|(Subscript)
|(Superscript)|(Footnote)|(Citation)|(XRef)|(Emphasis)|(OopsChar))">
<!-- Contents for other parameter entities.
These are all the varieties of lists.  -->
<!ENTITY % list.gp "((ItemizedList)|(OrderedList)|(VariableList)|(AdmonitionList)
)">
<!-- More elements for other parameter entities -->
<!ENTITY % literallayout.gp "((Example)|(LiteralLayout)|(Screen))">
<!-- More elements for other parameter entities -->
<!ENTITY % object.gp "((Graphic)|(Equation)|(Table)|(Admonition)|(BlockQuote)
|((%literallayout.gp;)))">
<!-- More elements for other parameter entities -->
<!ENTITY % component.gp "((Anchor)|(Para)|(Highlights)|(Epigraph)|((%list.gp;)
)|((%object.gp;))|(Oops)|(Comment)|(Figure)|(IndexTerm)|(Sidebar))">
<!-- Contents for the elements Para and BlockQuote -->
<!ENTITY % paracontent.gp "(((%inlinechar.gp;))|(Comment)|(Graphic)|(Equation)
)">
<!-- Contents for the element Example -->
<!ENTITY % examplecontent.gp "(Screen|LiteralLayout|ProgramListing)">
<!-- Contents for the element ListItem -->
<!ENTITY % listcontent.gp "((Para)|((%list.gp;))|(Figure)|((%object.gp;))|(Oops)
|(Comment))">
<!-- Contents for the element Admonition -->
<!ENTITY % admoncontent.gp "((Para)|((%list.gp;))|((%object.gp;))|(Oops)|(Comment)
|(IndexTerm))">
<!-- Elements comprising the body of a Table -->
<!ENTITY % tablecontent.gp "(TableSpec?, TableColHead*, TableRow+)">
<!-- Contents for the elements ProgramListing, LiteralLayout, Screen, and
RefSynopsisblock -->
<!ENTITY % literalcontent.gp "(((%inlinechar.gp;))|(Comment))">
```

```
<!-- Contents for the element Oops -->
<!ENTITY % oopscontent.gp "(((%inlinechar.gp;))|(Para)|((%list.gp;))|(Figure)
|((%object.gp;))|(Comment))">
<!-- Contents for the element Figure -->
<!ENTITY % figurecontent.gp "(((%object.gp;))|(Oops)|(Comment)|(IndexTerm))"
>
<!-- Contents for the element Sidebar -->
<!ENTITY % Sbcontent.gp "((Para)|((%list.gp;))|((%object.gp;)))">
<!-- Contents for the element Link -->
<!ENTITY % linkcontent.gp "(((%basechar.gp;))|(CiteBook)|(CiteChap)|(CiteSect)
|(Command)|(Filename)|(Flag)|(Function)|(Application)|(Hardware)|(Interface)
|(OopsChar)|(Subscript)|(Superscript))">
<!-- The headX.gp parameter entities handle nesting of heads.  -->
<!ENTITY % head1.gp "(((%component.gp;)+, (Head1|RefEntry)*)|(Head1|RefEntry)
+)">
<!ENTITY % head2.gp "(((%component.gp;)+, Head2*)|(Head2+))">
<!ENTITY % head3.gp "(((%component.gp;)+, Head3*)|(Head3+))">
<!ENTITY % head4.gp "(((%component.gp;)+, Head4*)|(Head4+))">
<!ENTITY % head5.gp "(((%component.gp;)+, Head5*)|(Head5+))">
<!-- Contents for the element IndexEntry -->
<!ENTITY % indexcontent.gp "(((%basechar.gp;))|(Command)|(Filename)|(Flag)|(Function)
|(Application)|(Hardware)|(Interface)|(CiteBook)|(CiteChap)|(CiteSect)|(OopsChar)
|(Emphasis)|(ComputerOutput))">
<!-- Contents for the element TitlePage -->
<!ENTITY % titlepagecontent.gp "((Author)|(Date)|(Copyright)|(ISBN)|(LegalNotice)
|(RevHistory)|(TitleBook)|(Editor)|(Edition)|(PrintHistory)|(Publisher)|(Series)
)">
<!-- Two parameter entities for the element IndexTerm, handling nesting -->
<!ENTITY % Secondarynest "((Tertiary)|(See)|(SeeAlso))">
<!ENTITY % Primarynest "((Secondary, (%Secondarynest;)?)|(See)|(SeeAlso))">
<!-- A parameter entity for the next parameter entity -->
<!ENTITY % refname.gp "(RefDescriptor?, RefName+, RefPurpose)">
<!-- Contents for the element RefNameDiv -->
<!ENTITY % refnamediv.content.gp "(((%refname.gp;))|(Comment)|(IndexTerm))"
>
<!-- Contents for some elements in RefEntry -->
<!ENTITY % refname.content.gp "(((%basechar.gp;))|(Command)|(Filename)|(Function)
|(Application)|(Hardware)|(Interface)|(OopsChar))">
<!-- Contents for the element RefHead1 -->
<!ENTITY % refdivbody.gp "(((%component.gp;)+, RefHead2*)|(RefHead2+))">
<!-- Document Structure -->
<!-- Base element -->
<!ELEMENT DocBook - - (Comment*, (%bookcomponent.gp;)+)>
<!-- Most of the book component elements -->
<!ELEMENT TitlePage - - ((%titlepagecontent.gp;)+)>
<!ATTLIST TitlePage id ID #IMPLIED>
<!ELEMENT Chapter - - (Title, TitleAbbrev?, (%head1.gp;))>
<!ATTLIST Chapter id ID #IMPLIED
Number CDATA #IMPLIED>
<!ELEMENT Preface - - (Title, TitleAbbrev?, (%head1.gp;))>
<!ATTLIST Preface id ID #IMPLIED>
<!ELEMENT ToC - - (Title, TitleAbbrev?, ToCentry1+)>
<!ATTLIST ToC id ID #IMPLIED>
<!ELEMENT LoT - - (Title, TitleAbbrev?, LoTentry+)>
```

```
<!ATTLIST LoT id ID #IMPLIED>
<!ELEMENT Appendix - - (Title, TitleAbbrev?, (%head1.gp;))>
<!ATTLIST Appendix id ID #IMPLIED
Number CDATA #IMPLIED>
<!ELEMENT Bibliography - - (Title, TitleAbbrev?, (%head1.gp;))>
<!ATTLIST Bibliography id ID #IMPLIED>
<!ELEMENT Glossary - - (Title, TitleAbbrev?, GlossaryEntry+)>
<!ATTLIST Glossary id ID #IMPLIED>
<!ELEMENT Index - - (Para*, IndexEntry+)>
<!ATTLIST Index id ID #IMPLIED>
<!ELEMENT Reference - - (Title, TitleAbbrev?, RefEntry+)>
<!ATTLIST Reference id ID #IMPLIED>
<!-- Most of the in-line elements -->
<!ELEMENT Comment - - ((%inlinechar.gp;)+)>
<!ELEMENT VarParam - - ((%varparam.content.gp;)+)>
<!ELEMENT Anchor - O EMPTY>
<!ATTLIST Anchor id ID #IMPLIED>
<!ELEMENT Command - - ((%basechar.gp;)+)>
<!ATTLIST Command RefEntry (Yes|No) No >
<!ELEMENT Filename - - ((%basechar.gp;)+)>
<!ATTLIST Filename RefEntry (Yes|No) No >
<!ELEMENT Flag - - ((%basechar.gp;)+)>
<!ELEMENT Application - - ((%basechar.gp;)+)>
<!ATTLIST Application RefEntry (Yes|No) No >
<!ELEMENT Function - - ((%basechar.gp;)+)>
<!ATTLIST Function RefEntry (Yes|No) No >
<!ELEMENT Hardware - - ((%basechar.gp;)+)>
<!ELEMENT Interface - - ((%basechar.gp;)+)>
<!ELEMENT CiteBook - - ((%cite.title.content.gp;)+)>
<!ELEMENT CiteChap - - ((%cite.title.content.gp;)+)>
<!ELEMENT CiteSect - - ((%cite.title.content.gp;)+)>
<!ELEMENT FirstTerm - - (#PCDATA)>
<!ELEMENT GlossaryTerm - - ((%glossterm.content.gp;)+)>
<!ELEMENT ComputerOutput - - ((%basechar.gp;)+)>
<!ELEMENT UserInput - - ((%uicontent.gp;)+)>
<!ELEMENT Subscript - - ((%script.content.gp;)+)>
<!ELEMENT Superscript - - ((%script.content.gp;)+)>
<!ELEMENT Quote - - ((%quote.content.gp;)+)>
<!ELEMENT Citation - - (#PCDATA)>
<!ELEMENT Emphasis - - (#PCDATA)>
<!ELEMENT OopsChar - - ((%inlinechar.gp;)+)>
<!ELEMENT WordAsWord - - (#PCDATA)>
<!ELEMENT SystemItem - - ((%basechar.gp;)+)>
<!ELEMENT Link - - ((%linkcontent.gp;)+)>
<!ATTLIST Link Target IDREF #IMPLIED
Type CDATA #IMPLIED>
<!ELEMENT FootnoteRef - O EMPTY>
<!ATTLIST FootnoteRef Target IDREF #IMPLIED
Mark CDATA #IMPLIED>
<!ELEMENT XRef - O (#PCDATA)>
<!ATTLIST XRef Target IDREF #CONREF
Type (Figure|Example|Table|Footnote|Chapter|Section|Sidebar)
#IMPLIED>
<!-- Contents for indexing terms in the text (not in the Index) -->
```

```
<!ELEMENT IndexTerm - - (Primary, (%Primarynest;)?)>
<!ATTLIST IndexTerm id ID #IMPLIED
Span (Start|End) #IMPLIED
Pagenum CDATA #IMPLIED>
<!ELEMENT Primary - - ((%indexcontent.gp;))>
<!ELEMENT Secondary - - ((%indexcontent.gp;))>
<!ELEMENT Tertiary - - ((%indexcontent.gp;))>
<!ELEMENT See - - ((%indexcontent.gp;))>
<!ELEMENT SeeAlso - - ((%indexcontent.gp;))>
<!-- Parts of a book component, sort of like objects, with some of the
elements that belong to them specifically -->
<!ELEMENT Head1 - - (Title, TitleAbbrev?, (%head2.gp;))>
<!ATTLIST (Head1|Head2|Head3|Head4|Head5)
id ID #IMPLIED
Number CDATA #IMPLIED>
<!ELEMENT Head2 - - (Title, TitleAbbrev?, (%head3.gp;))>
<!ELEMENT Head3 - - (Title, TitleAbbrev?, (%head4.gp;))>
<!ELEMENT Head4 - - (Title, TitleAbbrev?, (%head5.gp;))>
<!ELEMENT Head5 - - (Title, TitleAbbrev?, (%component.gp;)+)>
<!ELEMENT Footnote - - ((%component.gp;)+)>
<!ATTLIST Footnote id ID #IMPLIED>
<!ELEMENT Epigraph - - ((Para)+)>
<!ELEMENT Para - - ((%paracontent.gp;)+)>
<!ELEMENT Highlights - - (((%list.gp;)|Para|(Admonition))+)>
<!ATTLIST Highlights id ID #IMPLIED>
<!ELEMENT Graphic - - RCDATA>
<!ATTLIST Graphic id ID #IMPLIED
Type (GIF|TIFF|DVI|EPS|tbl|Tex|bitmap) GIF >
<!ELEMENT ItemizedList - - (ListItem+)>
<!ATTLIST ItemizedList Mark CDATA #IMPLIED>
<!ELEMENT ListItem - - ((%listcontent.gp;)+)>
<!ELEMENT OrderedList - - (ListItem+)>
<!ATTLIST OrderedList Numeration (Arabic|Upperalpha|Loweralpha|Upperroman|Lowerroman)
Arabic
Continuation (Yes|No) No >
<!ELEMENT VariableList - - ((Term+, ListItem)+)>
<!ELEMENT Term - - ((%inlinechar.gp;)+)>
<!ELEMENT AdmonitionList - - (ListItem+)>
<!ATTLIST AdmonitionList Heading CDATA #IMPLIED>
<!ELEMENT Admonition - - ((%admoncontent.gp;)+)>
<!ATTLIST Admonition Heading CDATA #IMPLIED>
<!ELEMENT Equation - - (Title?, TitleAbbrev?, Graphic)>
<!ATTLIST Equation id ID #IMPLIED>
<!ELEMENT Title - - ((%inlinechar.gp;)+)>
<!ATTLIST Title id ID #IMPLIED
Pagenum CDATA #IMPLIED>
<!ELEMENT TitleAbbrev - - ((%inlinechar.gp;)+)>
<!ELEMENT BlockQuote - - ((%paracontent.gp;)+)>
<!ELEMENT Example - - (Title?, TitleAbbrev?, (%examplecontent.gp;))>
<!ATTLIST Example id ID #IMPLIED>
<!ELEMENT ProgramListing - - ((%literalcontent.gp;)+)>
<!ELEMENT LiteralLayout - - ((%literalcontent.gp;)+)>
<!ELEMENT Screen - - (((%literalcontent.gp;)+) | (ScreenInfo, Graphic))>
<!ATTLIST Screen id ID #IMPLIED>
```

```
<!ELEMENT ScreenInfo - - (#PCDATA)>
<!ELEMENT Oops - - ((%oopscontent.gp;)+)>
<!ELEMENT Figure - - (Title, TitleAbbrev?, ((%figurecontent.gp;)+)*)>
<!ATTLIST Figure id ID #IMPLIED
Height CDATA #IMPLIED
Width CDATA #IMPLIED
Align (l|c|r) l
Float (YES|NO) NO >
<!ELEMENT Sidebar - - (Title, TitleAbbrev?, (%Sbcontent.gp;)+)>
<!ATTLIST Sidebar id ID #IMPLIED>
<!-- Table elements, provisional -->
<!ELEMENT Table - - (Title?, TitleAbbrev?, Graphic?, (%tablecontent.gp;))>
<!ATTLIST Table id ID #IMPLIED>
<!ELEMENT TableSpec - - (%inlinechar.gp;)>
<!ELEMENT TableColHead - - (TableCell+)>
<!ELEMENT TableCell - - ((%inlinechar.gp;)+)>
<!ELEMENT TableRow - - (TableCell+)>
<!ELEMENT GlossaryEntry - - (#PCDATA)>
<!ELEMENT IndexEntry - - ((%indexcontent.gp;)*)>
<!ATTLIST IndexEntry id IDREFS #IMPLIED>
<!ELEMENT LoTentry - - ((%inlinechar.gp;)+ )>
<!ATTLIST LoTentry Pagenum CDATA #IMPLIED>
<!-- Title Page elements -->
<!ELEMENT Author - - (Firstname+, Surname?)>
<!ELEMENT Firstname - - (#PCDATA)>
<!ELEMENT Surname - - (#PCDATA)>
<!ELEMENT Date - - (#PCDATA)>
<!ELEMENT Copyright - - (Year+, Holder*)>
<!ELEMENT Year - - (#PCDATA)>
<!ELEMENT Holder - - (#PCDATA)>
<!ELEMENT ISBN - - (#PCDATA)>
<!ELEMENT LegalNotice - - (Para+)>
<!ELEMENT RevHistory - - (Revision*)>
<!ELEMENT Revision - - (RevNumber, Date, AuthorInitials+, RevRemark)>
<!ELEMENT RevNumber - - (#PCDATA)>
<!ELEMENT AuthorInitials - - (#PCDATA)>
<!ELEMENT RevRemark - - (#PCDATA)>
<!ELEMENT TitleBook - - (Title, Subtitle?, TitleAbbrev?, ReleaseInfo?, BookAcronym?)
>
<!ELEMENT Subtitle - - (#PCDATA)>
<!ELEMENT ReleaseInfo - - (#PCDATA)>
<!ELEMENT BookAcronym - - (#PCDATA)>
<!ELEMENT Editor - - (Firstname+, Surname?)>
<!ELEMENT Edition - - (#PCDATA)>
<!ELEMENT PrintHistory - - (Para+)>
<!ELEMENT Publisher - - (#PCDATA)>
<!ELEMENT Series - - (#PCDATA)>
<!-- Contents for Tables of Contents -->
<!ELEMENT ToCentry1 - - ((%inlinechar.gp;)+, ToCentry2*)>
<!ATTLIST ToCentry1 Pagenum CDATA #IMPLIED>
<!ELEMENT ToCentry2 - - ((%inlinechar.gp;)+, ToCentry3*)>
<!ATTLIST ToCentry2 Pagenum CDATA #IMPLIED>
<!ELEMENT ToCentry3 - - ((%inlinechar.gp;)+, ToCentry4*)>
<!ATTLIST ToCentry3 Pagenum CDATA #IMPLIED>
<!ELEMENT ToCentry4 - - ((%inlinechar.gp;)+, ToCentry5*)>
```

```
<!ATTLIST ToCentry4 Pagenum CDATA #IMPLIED>
<!ELEMENT ToCentry5 - - ((%inlinechar.gp;)+)>
<!ATTLIST ToCentry5 Pagenum CDATA #IMPLIED>
<!-- All the elements for reference entries -->
<!ELEMENT RefEntry - - (Comment*, RefMeta, Comment*, RefNameDiv, RefSynopsis?,
RefHead1+)>
<!ATTLIST RefEntry id ID #IMPLIED>
<!ELEMENT RefMeta - - (RefFileName, RefSection, RefMiscInfo*)>
<!ATTLIST RefMeta id ID #IMPLIED>
<!ELEMENT RefFileName - - (#PCDATA)>
<!ELEMENT RefSection - - (#PCDATA)>
<!ELEMENT RefMiscInfo - - (#PCDATA)>
<!ELEMENT RefNameDiv - - ((%refnamediv.content.gp;)+)>
<!ATTLIST RefNameDiv id ID #IMPLIED>
<!ELEMENT RefDescriptor - - ((%refname.content.gp;)+)>
<!ELEMENT RefName - - ((%refname.content.gp;)+)>
<!ELEMENT RefPurpose - - ((%inlinechar.gp;)+)>
<!ELEMENT RefSynopsis - - ((RefSynopsisblock, RefHead2*)|(RefHead2+))>
<!ATTLIST RefSynopsis id ID #IMPLIED
Name (Synopsis|Syntax) Synopsis >
<!ELEMENT RefSynopsisblock - - ((%literalcontent.gp;)+)>
<!ELEMENT RefHead1 - - (Title, (%refdivbody.gp;))>
<!ATTLIST RefHead1 id ID #IMPLIED>
<!ELEMENT RefHead2 - - (Title, (%component.gp;)+)>
<!ATTLIST RefHead2 id ID #IMPLIED>
]>
```