Guide to the DocBook DTD

HaL Computer Systems International, Ltd.

and

O'Reilly & Associates, Inc.

DTD release 1.2.2
Documentation version 1.0 for release 1.2.2
17 May 1993

Release 1.2.2 and documentation copyright 1992, 1993 HaL Computer Systems International, Ltd., 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.

A version of the canontbl.dtd, which is copyright SoftQuad, Inc., Toronto, is included in the DocBook DTD by kind permission of SoftQuad.

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

Table of Contents

Acknowledgements iii
How to Get the DocBook DTD Online iii
Development of the DocBook DTD 1
A Structural View of the DTD 1
Sets, Books, and Their Parts 2
- Preface, Chapter, Appendix, Bibliography, Glossary, and Index 4
- Table of Contents 6
- Lists of Titles (LoT) 6
- DocInfo 6
- References 6
- Sections (Heads) and Tasks 7

Block-Oriented Elements 7
- Paragraphs 8
- Lists 9
- Tables 12
- Figures and Equations 14
- Examples, LiteralLayouts, ProgramListings, and Screens 14
- BlockQuotes 15
- Sidebars 15
- Footnotes 15
- Admonitions and other Blocks 16

Elements Indicating Nonlinear Relationships 16
- Comments 16
- IndexTerms 16
- IndexEntries 17
- Links, ExternalLinks, and Anchors 17
- Cross References (XRefs) 18

In-line Elements 18
Usage of In-line Elements 18
Reference Entries 18
- RefEntry Basics 18
- RefEntry Variations 19
- RefMeta 20
- RefNameDiv 21
- RefSynopsisDiv 21
- RefHeads 22

Miscellaneous 22
The Elements Alphabetized 23

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 1993) 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 and related ASCII files may be found in a file named "docbook.*N*.shar", where *N* is the current revision number:

```
ftp> get docbook.1.2.2.shar
```

Most of these files also exist separately and may be ftp'd individually.

The "get" command will put this ASCII shar file on your system. You must later "unshar" it on your system: **unshar docbook.1.2.2.shar.**

Back to your ftp session. The DocBook documentation is in 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 or viewer.

Development of 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 documents. It describes the structures we and others have encountered in processing several large bodies of documentation. 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 it for their domain of applications. Anyone making such changes should read the section near the top of the DTD itself regarding local modifications.

We are committed to reviewing revisions made to the DTD by various groups and evaluating them for inclusion in the next release of the DTD.

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

A Structural View of the DTD

Four kinds of elements are defined in the DocBook DTD There are elements that indicate the hierarchy of the information. The root of this hierarchy is intended to be the DocBook element, which has no semantic meaning. The elements Set, Book, and Part are provided for representing large document structures. A Set contains Books, and a Book may be divided into Parts. Either Books or Parts contain book components, such as Chapters or Appendices. Below these book components are additional levels of sectioning, marked as Sect1 through Sect5.

A section is further organized by block-oriented elements such as Paras (paragraphs). Other block-oriented elements, 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.

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.

This DTD also provides many in-line elements, some of which are peculiar to computer documentation. These in-line elements allow you to label specific terms or usage. Both general and specific elements are provided.

The following sections discuss how the syntactic elements defined in the DocBook DTD may be combined, using examples of text tagged in SGML. Following them is a section that provides brief definitions of all the elements.

Please note:

- Element names in SGML are not case-sensitive. The capitalization used here is merely conventional
- In the examples that follow, some punctuation and mathematics characters are represented by external 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.
- This DTD is generalized, so as to be applicable to a wide range of document designs. It is descriptive rather than prescriptive, and thus allows structures that no one would want to see in a finished document. It is the responsibility of writers and editors to determine that their documents are appropriately designed. That task might be aided by constructing more prescriptive versions of this DTD, tailored to specific document designs.

Sets, Books, and Their Parts

The intended base document element for the DocBook DTD is DocBook. This element encloses all other markup in the document. Within the DocBook tags you can make Books that contain various book components. Here's a schema of a fairly vanilla Book:

Book
DocInfo
ToC
LoT
Preface
Chapter
Chapter
Chapter
Chapter
Reference
Appendix
Appendix
Glossary
Bibliography
Index

Books may have Chapters and Appendices grouped in Parts:

Book
DocInfo
Preface
Part
PartIntro

Chapter
Chapter
Part
PartIntro
Chapter
Chapter
Chapter
Part
PartIntro
Chapter
Chapter

Another example show how a reference manual might be arranged:

Book
DocInfo
ToC
LoT
Preface
Reference
Reference
Reference
Reference
Appendix
Appendix
Appendix
Appendix
Index

A Set of Books may have a Set-level DocInfo and must contain two or more Books:

Set
DocInfo
Book
Book
Book

The DocBook DTD does not require that a valid document be an entire book. Any of the following can also be a valid document on its own, within DocBook tags: Appendix, Bibliography, Book, Chapter, DocInfo, Glossary, Index, LoT (List of Titles), Part, PartIntro, Preface, RefEntry, Reference, and ToC (Table of Contents). DocBooks may contain other DocBooks. Thus one may make a DocBook document that includes three Chapters, each wrapped in its own DocBook tags, which are shown verbatim rather than schematically:

<DOCBOOK>
<DOCBOOK>
Chapter
</DOCBOOK>

```
<DOCBOOK>
  Chapter
</DOCBOOK>
<DOCBOOK>
  Chapter
</DOCBOOK>
</DOCBOOK>
```

If the part of a DocBook are also DocBooks, as in this example, any relationship among them that may be specified by the DTD (for example, that within a Book, Chapters appear before Appendices) is ignored.

A Book must have a DocInfo and may have a ToC, followed by any number of LoTs, followed by any number of Prefaces; it must then have one or more Parts or one or more Chapters, and may conclude with any number of the following, in order: Reference, Appendix, Glossary, Bibliography, and Index.

A Part must have a Title, may have a TitleAbbrev, may have a PartIntro, and must contain two or more Chapters or Appendices; or it may consist of one or more References.

We suspect that book components will often be used individually. Here is a Chapter standing on its own, schematically and in actual markup:

```
Chapter
  Title
  TitleAbbrev
  Para
  Para
```

```
<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>
```

For best results be sure to start the text of an element immediately after the start tag, without a space or line break. Your SGML application may treat a line break as significant data.

Preface, Chapter, Appendix, Bibliography, Glossary, and Index

A Chapter or Appendix may have a DocInfo, must have a Title and some text—at least a Sect, a Para, a list, or a RefEntry—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 document.

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."

Bibliography, Glossary, and Index are allowed to have more specialized contents also: BiblioEntry for Bibliography, GlossaryEntry for Glossary, and IndexEntry for Index. These specialized book parts may be divided into BiblioDivs, GlossDivs, or IndexDivs, respectively. (Please note that an IndexEntry is part of an Index, whereas an IndexTerm is part of the text of a document, marking the term to be indexed.) Additionally, an IndexDiv may have a SegmentedList, to accomdate permuted indices. A Glossary may end with a Bibliography.

IndexEntries and IndexTerms are discussed below, *s.v.* Index.

BiblioEntries may begin with a TitleAbbrev, then must contain one or more of Author, CorpAuthor, Editor, and Title; these may be followed by optional Subtitle, Date, Publisher, and plain text. As BiblioEntry has the notorious SGML "mixed content," you must begin the contents of the entry immediately after its start tag, without a line break.

GlossaryEntries are simply pairs of GlossaryTerms and GlossaryDefs, both of which may contain in-line elements.

Bibliography, Glossary, Reference, and Index may be book components on their own or may occur at the beginning or end of a Preface, Chapter, Appendix, or Sect. A RefEntry (the element for a man page) cannot be a book component, but can appear in the same places within Preface, Chapter, Appendix, or Sect as can a Bibliography, Glossary, or Index. You could construct a Chapter this way:

Chapter
- **Title**
- **Para**
- **Sect1**
  - **Title**
  - **Para**
  - **Para**
  - **Para**
  - **Glossary**
  - **Bibliography**
- **Sect1**
  - **Title**
  - **RefEntry**
  - **RefEntry**
  - **RefEntry**
  - **RefEntry**
  - **Index**

Table of Contents

A ToC may have a Title and 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 LINKEND="S1-061-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.

DocInfo

A DocInfo contains metainformation about the document to which it is attached, which may be a Set, Book, or book component. It may contain any number of any of these elements, in any order: Author, AuthorInitials, BookTitle, Copyright, CorpAuthor, CorpName, Date, Editor, Edition, InvPartNumber, ISBN, LegalNotice, OrgName, PrintHistory, ProductName, ProductNumber, Publisher, PubsNumber, ReleaseInfo, RevHistory, Series, Subtitle, and VolumeNumber.

References

A Reference is a collection of RefEntries. Before its RefEntries, it has an optional DocInfo, a required Title, an optional TitleAbbrev, and an optional PartIntro. It may be a book component or may occur at the beginning or end of a Preface, Chapter, Appendix, or Sect.

Reference
Title
PartIntro

RefEntry
RefEntry
RefEntry
RefEntry
RefEntry

Sections (Heads) and Tasks

A Preface, Chapter, or Appendix may contain up to five levels of hierarchical sections called Sect1, Sect2, etc. In a RefEntry, these are RefSect1 and RefSect2 (no further subheads). Headings be nested properly: a Sect3 is not allowed directly under a Sect1 without a Sect2 in between. Sects1–4 that contain lower-level Sects may be empty of other content.

This DTD allows Paras, lists, and other block-oriented elements to be inserted between Sects, although the use of these elements in such a fashion is widely deprecated. Paras may be titled, thus providing for bridging heads. Otherwise, a section includes a Title (the text 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 (DocInfo, Glossary, etc.) or another section of the same level.

A section must have a Title. Thus,

```
<SECT1 ID="S1-061-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>
</SECT1>
```

Sections have a Number attribute, in case you wish to hard-code section numbering.

Within a book component or Sect, one may mark off any series of Paras and lists as a Task, which may have a Title and TitleAbbrev. Task is not a hierarchical element, and may not itself contain Sects.

Block-Oriented Elements

Most book components can contain:

- Highlights (list of main points)
- an Epigraph
- Paragraphs
- Sections (Headings)
- Lists
- Admonitions (Warnings, Tips, and so on)

- Tables, Figures, Equations, BlockQuotes
- Examples, LiteralLayouts, Screens
- Graphics
- Sidebars
- Comments
- Terms to be indexed
- Lists
- Tables
- Figures
- LiteralLayouts
- Examples
- Screens
- BlockQuotes
- Footnotes

Highlights (a section of major points appearing in the book component) and an Epigraph (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 in-line elements (sentences within paragraphs are not tagged, for example). The DocBook DTD does not distinguish among different types of paragraphs: indented, nonindented, continuation, and hanging. Paras may contain block-oriented elements.

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.

```
<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>
</WARNING>
```

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 in- line elements discussed below in the "In-line Components" section.

Lists

Lists may be of five types:

- ItemizedList (bulletted, dashed, etc.)
- OrderedList (numbered or lettered)
- VariableList (composed of paragraphs or sets of paragraphs, etc., with associated Terms)
- SegmentedList (a series of entries composed of equal numbers of segments, with optional titles pertaining to all the first segments, all the second segments, and so on)
- SimpleList (composed of terms that may be arranged in columns or in-line)

Lists may be nested within ListItems (an ItemizedList inside a ListItem of an OrderedList, for example), and ListItems may include Paras and other block-oriented elements.

ItemizedLists use an attribute, Mark, to determine the character that precedes each ListItem. You must supply a word to specify the mark that should appear before each ListItem, such as Bullet, Dash, Box, or Check. You might want to use a character entity instead (such as •); there is no default. Here's 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 connec-
        tion could not be made.
        </PARA>
    </LISTITEM>
    <LISTITEM>
        <PARA>Get information about the physical screen, and use it to calculate the de-
        sired 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>
```

Normally an ItemizedList nested within a list of the same type is given a different mark or alphanumeric style (as in an outline). 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.

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. Finally, OrderedList has an InheritNum attribute, the values of which are Inherit and Ignore, with Ignore as the default. Set InheritNum to Inherit if you wish nested lists to pick up the numbering of the ListItem within which they are nested (so that in an OrderedList nested under the third ListItem of another OrderedList the numbering would be 3.1, 3.2, 3.3, instead of 1, 2, 3).

```
<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 ap-
        pears 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, which may have a Title and TitleAbbrev, is otherwise composed of VarListEntries, each of which in turn is composed of a set of one or more Terms followed by a ListItem. Terms may contain in-line elements and Synopses (for Synopses see RefEntries, below). Schematically:

VariableList
- **VarListEntry**
  - **Term**
  - **Term**
  - **ListItem**
    - **Para**
    - **Para**
- **VarListEntry**
  - **Term**
  - **ListItem**
    - **Para**
- **VarListEntry**
  - **Term**
  - **ListItem**
    - **Para**

In actual markup:

```
<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 XOpenDisplay.
        </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>
```

A SegmentedList is a list composed of labelled sets of information, and may be used to represent information often presented in simple tables. A SegmentedList may have a Title and TitleAbbrev, followed by any number of SegTitles and two or more SegListItems. SegListItems are composed of two or more Segs, which may contain in-line elements.

```
<SEGMENTEDLIST>
    <SEGTITLE>Museum
    </SEGTITLE>
    <SEGTITLE>City
    </SEGTITLE>
    <SEGTITLE>Collections
    </SEGTITLE>
    <SEGLISTITEM>
        <SEG>Asian Art Museum
        </SEG>
        <SEG>San Francisco
        </SEG>
        <SEG>East Asian, Indian, and Islamic
        </SEG>
    </SEGLISTITEM>
    <SEGLISTITEM>
        <SEG>British Museum
        </SEG>
        <SEG>London
        </SEG>
        <SEG>all areas
        </SEG>
    </SEGLISTITEM>
    <SEGLISTITEM>
```

```
                <SEG>Louvre
                </SEG>
                <SEG>Paris
                </SEG>
                <SEG>mostly paintings
                </SEG>
            </SEGLISTITEM>
</SEGMENTEDLIST>
```

Such a SegmentedList might appear in print as follows:

Museum:	Asian Art Museum
City:	San Francisco
Collections:	East Asian, Indian, and Islamic
Museum:	British Museum
City:	London
Collections:	all areas
Museum:	Louvre
City:	Paris
Collections:	mostly paintings

SimpleList is intended for long lists of single words or short phrases. It consists of one or more Terms, and has a Wrap attribute, the value of which may be Vert, Horiz, or Inline. Vert and Horiz indicate that the Terms should be displayed in columns; if Vert, the Terms should be ordered from top to bottom of each column, and if Horiz, the Terms should be ordered left to right, one row at a time. Inline indicates that the Terms should be displayed in-line, such as: `a, b, c, d, e, f, g, h, i, j, k,` and `l.` If these Terms are wrapped vertically, they are displayed in an array such as this:

```
a   d   g   j
b   e   h   k
c   f   i   l
```

If the Terms are wrapped horizontally:

```
a   b   c   d
e   f   g   h
i   j   k   l
```

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 another form. The DocBook DTD thus allows you to mark up a table in SGML and also provide another representation of it in an external file. Table allows two alternate content models. The first is our own invention; the second is a version of the canontbl.dtd, copyright SoftQuad, Inc. (Used by kind permission of Yuri Rubinsky,

SoftQuad.) The first model is described here; for the second, which you may find useful to use with the Author/Editor product, see SoftQuad's product documentation.

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), TableColHeads (a row of headings above each column), and TableRows of TableCells. Table has a Number attribute.

Graphic may contain graphical data or point to exterior files that contain such data. It has an attribute, Format, that indicates how that data is formatted. Format may have the value GIF, TIFF, CGM, DVI, EPS, TBL, EQN, PS, DITROFF, or PIC. There is no default. Graphic may point to an external file via its Fileref or Entityref attributes. The value of Fileref should be a filename, qualified by a pathname if desired; the value of Entityref must be an external entity, such as &chap3fig4;.

Here is a sample Table:

```
<TABLE ID="TB-061-ch05-1">
    <TITLE>Geometry Specification x and y Offsets
    </TITLE>
    <GRAPHIC FORMAT=" tbl" FILEREF="tbl/ch05.01.tbl">
    </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 and Equations

A Figure must have a Title, may have a TitleAbbrev, and may contain one or more of the following, not all of which are appropriate for inclusion in a Figure: BlockQuote, Caution, Equation, Example, Footnote, Graphic, Important, LiteralLayout, Table, Note, ProgramListing, Screen, Synopsis, Task, Tip, Warning, ExternalLink, FootnoteRef, IndexTerm, Link, and XRef.

Figure has 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 an SGML application, you need supply no values for any of these attributes. (In the following example an Entityref attribute is used instead of the Fileref attribute seen in the previous example.)

```
<FIGURE ID="FG-061-ch05-2" ALIGN="C" FLOAT="YES">
    <TITLE>The xdmlogwin Display
    </TITLE>
    <GRAPHIC ENTITYREF="&gr061ch05t2; ">
    </GRAPHIC>
</FIGURE>
```

An Equation may have a Title and TitleAbbrev, and then must contain a Graphic, which may include the equation data or may refer to an exterior file in the manner described above.

Examples, LiteralLayouts, ProgramListings, and Screens

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

A Example is a program listing or a segment of one. An Example may have a Title and a TitleAbbrev. Following the Title, if any, an Example may contain any combination of Screen, LiteralLayout, and ProgramListing.

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
    <LITERALLAYOUT>
    Butros Butros Ghali
    Secretary General
    United Nations
    </LITERALLAYOUT>
about the Security Council's jurisdiction over the Moon.
</PARA>
```

A ProgramListing is just like a LiteralLayout, but should be reserved for listings of programs, which may require a different style of presentation.

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 text and in-line elements, in the manner of 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:

```
<SCREEN ID="SC-1015-12-24"> <PROMPT>%</PROMPT> <USERINPUT>make install</USERINPUT>
install -c -m 0644 rgb.txt /usr/lib/X11 install -c -m 0644 rgb.dir /usr/lib/X11 install -c -m
0644 rgb.pag /usr/lib/X11 install -c -s showrgb /usr/lib/X11 install in ./rgb done </SCREEN>
```

Here's a Screen with graphical content:

```
<SCREEN>
    <SCREENINFO> Press the F1 key and type "catfile" while holding down the Alt, Page
    Down, and F12 keys.
    </SCREENINFO>
    <GRAPHIC FORMAT="GIF" FILEREF="figs/05.07.gif;">
    </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 may have a Title and a TitleAbbrev, and must have text in the form of Paras, Lists, Figures, or Tables; no Heads are allowed.

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 Linkend 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 *). Footnote also has a Number attribute, should you wish to hard-code the numbering of footnotes.

Admonitions and other Blocks

Admonitions are those special sections of text in which a documentation writer calls attention to some matter of importance. Admonitions in the DocBook DTD take the form of the elements Caution, Note, Warning, Tip, and Note. All may be titled, and may contain Paras and other block-oriented elements.

Abstract, a summary of a document, and AuthorBlurb, a section of informattion on a document's author, may be titled and may contain Paras and other block-oriented elements.

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. Beware! Just tagging text as a Comment will not ensure that it is hidden; you must arrange that in your application.

IndexTerms

IndexTerms are words or phrases to be indexed. 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 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 print.

Should you wish to tag terms for indexing where they lie in the text you may wish to specify a different string to appear in the index (the text says "to rotate your terminal" but the index should say "rotating your terminal"). In this case you should provide that string within Primary, Secondary, or Tertiary, tagging it as IndexAs.

```
<PARA>This paragraph deals with two subjects that should be listed in the index: how to
rotate your monitor and how to adjust its height.
</PARA>
<INDEXTERM>
    <PRIMARY>rotating your monitor
    </PRIMARY>
</INDEXTERM>
<INDEXTERM>
    <PRIMARY>monitor
    </PRIMARY>
        <SECONDARY>rotation of
        </SECONDARY>
    </PRIMARY>
```

```
</INDEXTERM>
<INDEXTERM>
    <PRIMARY>monitor
    </PRIMARY>
        <SECONDARY>adjustment of
        </SECONDARY>
            <SEEALSO>troubleshooting
            </SEEALSO>
    </PRIMARY>
</INDEXTERM>
```

IndexEntries

IndexEntries, which occur in the Index, might be constructed by extracting and modifying the items in the text that have been tagged for indexing with IndexTerm and its allowable subelements Primary, Secondary, Tertiary, See, and SeeAlso. In the process one would simply drop IndexAs tags while retaining their content.

IndexEntries are put together just like IndexTerms, with the corresponding elements PrimaryIE, SecondaryIE, TertiaryIE, SeeIE, and SeeAlsoIE.

Links, ExternalLinks, and Anchors

A Link is a pointer to another section of a document, or another document. It may include in-line elements as text and has a required Linkend attribute, which is the ID of the element linked to. It also has an Endterm attribute, the value of which is the ID of an element the text of which is to appear as the text of the Link, such as the label of the hot spot a user clicks on to use the Link in an online display:

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

or, to link to a section and provide its title as the displayed text:

```
<LINK LINKEND="S1-061-ch05-1" ENDTERM="T1-061-ch05-1"> Click here
</LINK>
```

Link uses the SGML mechanism of IDREF in pointing to an element by its ID. An SGML application may report as an error an IDREF to an ID that does not occur in the document being processed. While this may be harmless, another element, ExternalLink, has been provided in parallel to Link. ExternalLink does not use IDREF, and may thus be employed to refer to IDs outside the document without generating errors.

Anchor is an in-line element that marks a location as a Linkend for a Link. 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 Linkend and Endterm attributes, just like Link. An XRef may have plain character content, or no content, but both start and end tags must be supplied.

> See <XREF LINKEND="S1-061-ch05-1" ENDTERM="T1-061- ch05-1">, for more information.

This references the section that has the ID S1-061-ch05-1 and supplies the text of its Title, which has the ID T1-061-ch05-1, in the manner of: "See *Terminal Emulation and the xterm Terminal Type* for more information."

In-line Elements

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

All of the elements defined in the DTD in the parameter entities inlinechar.gp and paracontent.gp, may appear in-line; the names of these elements are also marked in the alphabetical list of elements.

Synopsis may appear in-line, so that synopses can be marked wherever they are discussed.

Usage of In-line Elements

Trying to determine the most specific in-line element that can be applied to the specialized terms in your document can be time-consuming and frustrating. Instead, you may wish to tag these terms using only the most general of the in-line elements. We have found it useful, in converting files from troff to SGML, to translate the font changes that mark off these terms in print into five DocBook tags: Emphasis, Literal, VarParam, UserInput, and Interface (which correspond well to our conventional use of italic, typewriter font, italic typewriter, bold typewriter, and sans serif).

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 occur within a Chapter or Appendix, or a collection of RefEntries may be made into a book component called Reference.

RefEntry Basics

A RefEntry must begin and end with RefEntry tags, and may be included in most book components. It may be processed separately when wrapped in DocBook tags, but may not be a book component itself.

A RefEntry contains, in order: one DocInfo (optional); one RefMeta (required); one or more Comments, ExternalLinks, FootnoteRefs; IndexTerms, Links, or XRefs (optional); one RefNameDiv (required); one RefSynopsisDiv (optional); one or more RefSect1s (required). Schematically, with additional elements particular to RefEntries:

RefEntry
- **DocInfo**
- **RefMeta**
  - **RefFileName**
  - **ManVolNum**
  - **RefMiscInfo**
  - **RefMiscInfo**
- **RefNameDiv**
  - **RefName**
  - **RefPurpose**
- **RefSynopsisDiv**
  - **Synopsis**
- **RefSect1**
  - **RefSect2**
  - **RefSect2**
- **RefSect1**
- **RefSect1**

Comments, ExternalLinks, FootnoteRefs, IndexTerms, Links, or XRefs may also appear at the end of RefNameDiv.

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, ManVolNum, RefMiscInfo, RefNameDiv, RefDescriptor, RefName, RefPurpose, RefSynopsisDiv, RefSect1, and RefSect2.

RefMeta

The RefMeta contains, in order, a RefFileName, an optional ManVolNum, and any number of 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. ManVolNum 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 ManVolNum; 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. Other examples of RefMiscInfo contents are copyright, date, draft status, hardware architecture, and a descriptive phrase for use in a print header. All of these RefMeta elements may contain only text, without further nested elements.

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>
    <MANVOLNUM>ldsysdump(1)
    </MANVOLNUM>
    <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, which may be followed by any number of RefClasses. In RefClass one may indicate the applicability or scope of the topic of a RefEntry (such as the element SWapplic or a string such as "LocalBlocking"). After the RefClasses, RefNameDiv may contain any number of Comments, ExternalLinks, IndexTerm, Links, and XRefs. RefName may contain further nested elements from the parameter entity computerterms.gp, such as Command or Function. RefPurpose may contain any in-line elements.

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>
```

RefSynopsisDiv

RefSynopsisDiv 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 RefSynopsisDiv 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 RefSynopsisDiv may be a Synopsis (containing a single line or a set of lines), or a Synopsis followed by any number of RefSect2s, or one or more RefSect2s.

Here is a valid RefSynopsisDiv:

```
<REFSYNOPSISDIV ID="RS-061-66">
    <SYNOPSIS>
        <COMMAND> ls </COMMAND>
        <OPTIONAL><VARPARAM> options</VARPARAM></OPTIONAL> <FILENAME>
        <VARPARAM>directory_name</VARPARAM><FILENAME>
    </SYNOPSIS>
</REFSYNOPSISDIV>
```

RefHeads

The rest of a reference page consists of one or more divisions, which are to be tagged as RefSect1s, which may have RefSect2s. These are just like the Sect1 and Sect2 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 DocInfo, 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.