Single-Source Indexing

Jan C. Wright
Wright Information Indexing Services, Inc.
P.O. Box 658
Sandia Park, NM 87047
505-281-2600
jancw@wrightinformation.com

ABSTRACT
As more and more content is being produced and distributed in multiple formats, the issue of providing indexes for various formats becomes important. Indexes can be considered as residing within interfaces, rather than just more pages accompanying a printed piece, or a tab stuck in a online help file. Designing indexes that work equally well in any of the various interfaces a document may be displayed in, whether print, help, PDF, or on the Web, presents a real challenge.

Understanding the structure and relationships available in each destination format allows the indexer to design the index to work well in each instance. Ignoring an output format or assuming that the index is a simple construction leads to poorly designed online indexes, in which one format's requirements have been sacrificed for the output needs of another. In addition, print indexes do not translate well to online without consideration of screen design and user behavior.

In this paper we discuss the interface indexing design issues for print, online HTML Help, PDF, XML, and plain HTML.

Categories and Subject Descriptors

General Terms
Documentation

Keywords
Indexing, Single-sourcing

1. INTRODUCTION
As more and more people try to get one index to work in all the content they produce from a base set of files, whether print or online, it becomes evident that indexes are not just another piece of content to be converted to another format. Since they function as search tools, they should change their functionality to best utilize the interface they are residing in.

Taking base files for a documentation set and creating both print and online output often poses difficulties in terms of the main content of a piece. When the process is ready to convert the indexing for the files, a whole new set of difficulties comes to light. The indexing often doesn't work correctly online, or all of the information that was represented by page ranges falls away with only a single link to represent it online. Links may be dead, missing, presented oddly in the interface, or too long to be visible in the screen setting.

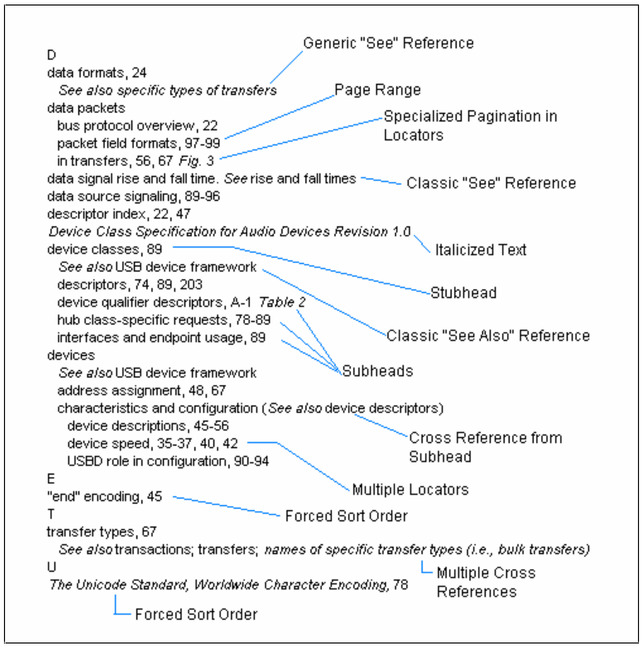
Online indexing interfaces can range from displaying embedded key terms from a thesaurus in neatly-defined fields in databases, to a CD-ROM index with pictures of dinosaurs beside each letter section, to a Web-based HTML index jumping to other sites, to the Help file index that comes with a software product. The display of each online index varies widely, and the techniques and tools used to index the content need to be analyzed to meet that display's particular needs.

Indexes, by their very nature, are not straightforward pieces of content. They have a database nature to their construction that is often not considered until they undergo a conversion process. Indexes consist of records which are compiled into a readable, searchable format. That underlying database structure must be constructed and then converted with an eye towards creating a usable tool in multiple output formats. Writing indexes means writing a readable, comprehensible piece that happens to reside in a set of records that gain meaning when they are compiled together in an interface. This is a unique challenge when single-sourcing files.

Once we realize that we are working with a search tool in multiple interfaces, one that is based on records in a database, we can start designing the indexing with an eye towards requirements, tradeoffs, and usability. To understand what is required, it's best to start by looking at the elements that make up indexes, since some specific fields in the database cause problems when an index is single-sourced. Then let's look at samples of each kind of interface that single-sourced documents are compiled into, so that we can understand how to identify a list of requirements for single-sourcing indexes.

1.1 Identifying Index Elements

A typical segment of print indexing can reveal a lot of index elements for us:



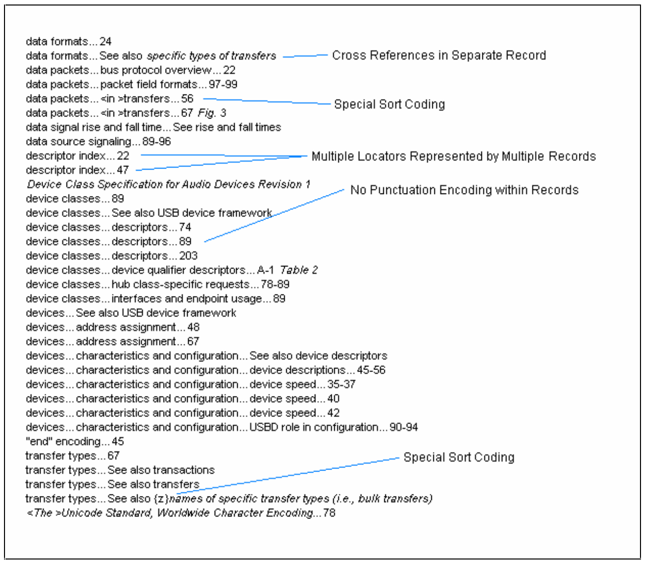
D
data formats, 24
  See also specific types of transfers
data packets
  bus protocol overview, 22
  packet field formats, 97-99
  in transfers, 56, 67 *Fig. 3*
data signal rise and fall time. *See* rise and fall times
data source signaling, 89-96
descriptor index, 22, 47
Device Class Specification for Audio Devices Revision 1.0
device classes, 89
  See also USB device framework
  descriptors, 74, 89, 203
  device qualifier descriptors, A-1 *Table 2*
  hub class-specific requests, 78-89
  interfaces and endpoint usage, 89
devices
  See also USB device framework
  address assignment, 48, 67
  characteristics and configuration (*See also* device descriptors)
    device descriptions, 45-56
    device speed, 35-37, 40, 42
    USBD role in configuration, 90-94
E
"end" encoding, 45
T
transfer types, 67
  See also transactions; transfers; *names of specific transfer types (i.e., bulk transfers)*
U
The Unicode Standard, Worldwide Character Encoding, 78

Figure 1: Index Elements

Figure 1 shows a typical segment of a compiled print index. There's no visual evidence of the database in which these records reside, just a finished search tool. This print interface, because print really IS an interface for presenting information, has come to represent the standard idea of what an index consists of.

- There are main entries, such as "data source signaling."
- There are entries that require more elaboration, or need to be broken down, such as the subheads following "device classes."
- Subheads can be second levels, such as under "device classes," or go to third levels in a deep index, such as the entries under "devices."
- Usually unnoticed in the main head-subhead groupings is the entry for "device classes" itself, the one with the locator on page 89. These standalone entries have come to be called "stubheads," since they create a stubbed heading in indexes. These are headings that will be created by the compiler if needed, or can be created by assigning an entry with no subheads such as the one on page 89, and in print are often used to indicate the main discussion. In general, they function as a placeholder in the compiled/printed version of the index in order to display the main heading. Some tools require them to have a locator, and some do not. Whether or not a stubhead record with no subheads has been entered, the compiler creates the main head for the subheads below. (See section 2.4.)
- There are the portions of the entries that represent where the material is to be found, the locators. In this example, they are page numbers, but they can also be topic links.
- Some topics are found in multiple places in the content, and are represented by multiple locators, such as those following "descriptor index."
- Some topics are long, and the index should let the user know that this is a long discussion, covering several pages. In these cases, a page range is used, such as the one following "data source signaling."
- Locators may represent different numbering systems present in the piece, such as the one for "data packets: in transfers." The compiler must be able to handle the ordering of locators, so that they are presented logically to the user.
- Cross references, in several varieties, are used. "See" references direct a reader from an unused term to a preferred term, in the hopes that the reader will find the time to go look in another place within the index, the place where all the entries are hopefully spelled out. These are only used when it is worthwhile to make the user work harder. Duplication of entries under both heads is usually easier on the user, but space constraints can lead to using "See" references.
- "See also" references give clues to related material that the reader might also be interested in.
- Generic "See" references are those which refer the user to a type of entries to be consulted. No one specific entry is singled out. These are used when you have a lot of entries of a particular type, such as commands or dialog boxes.
- Cross references that are attached to a subhead are a fourth kind of reference.
- Sorting codes, coding attached to the record to force it to sort in a different way than the machine's ASCII default, are needed to have entries such as "The Unicode Standard…" come up in the U's instead of in the T's section. These codes are usually processed with the record. You can also see forced sort order coding results within the subheads, where the word "in" is being ignored under "data packets"

This index construction was created as a set of database records, and in order to look at single sourcing indexes, it's important to think about the underlying database structure. Figure 2 shows the records that created this chunk, so that the important pieces can be identified in their original state before compilation.



```
data formats...24
data formats...See also specific types of transfers
data packets...bus protocol overview...22
data packets...packet field formats...97-99
data packets...<in >transfers...56
data packets...<in >transfers...67 Fig. 3
data signal rise and fall time...See rise and fall times
data source signaling...89-96
descriptor index...22
descriptor index...47
Device Class Specification for Audio Devices Revision 1
device classes...89
device classes...See also USB device framework
device classes...descriptors...74
device classes...descriptors...89
device classes...descriptors...203
device classes...device qualifier descriptors...A-1 Table 2
device classes...hub class-specific requests...78-89
device classes...interfaces and endpoint usage...89
devices...See also USB device framework
devices...address assignment...48
devices...address assignment...67
devices...characteristics and configuration...See also device descriptors
devices...characteristics and configuration...device descriptions...45-56
devices...characteristics and configuration...device speed...35-37
devices...characteristics and configuration...device speed...40
devices...characteristics and configuration...device speed...42
devices...characteristics and configuration...USBD role in configuration...90-94
"end" encoding...45
transfer types...67
transfer types...See also transactions
transfer types...See also transfers
transfer types...See also {z}names of specific transfer types (i.e., bulk transfers)
<The >Unicode Standard, Worldwide Character Encoding...78
```

Cross References in Separate Record
Special Sort Coding
Multiple Locators Represented by Multiple Records
No Punctuation Encoding within Records
Special Sort Coding

Figure 2: Index Records

As you can see, in the records themselves, each main head is repeated as the first element, the first "field," in the complex entries with subheads. The stubhead entry is a separate entry if it has been assigned a locator. If not, the compiler creates it. Cross references also require their own records. Each locator that is picked up requires its own record as well. (Note: in some indexing tools such as Cindex or Macrex, several locators can be combined in a single record. In most embedded tools, this doesn't work, since the entries are placed on separate pages and must be represented as duplicated records.)

Also note that the formatting of italics on the cross references is not present in this particular record sample. Formatting is something that the compiler (Cindex in this case) applies to these records in the particular tool these records were generated in. The most valuable indexing tools need to provide for different formatting styles without having to pre-format the content of individual records.

Already, you can see that getting the basic records into a compiled format just for print requires some processing – compiling the subheads under "device classes," and formatting the sections correctly, knowing that the stubhead for "device classes" needs to come first, applying italics on all the cross references, and also placing them where they make the most sense.

The best indexing programs for print indexing (and for single-sourcing) leave all the formatting to the compiler, so that it is easy to change punctuation between entries in the compile, change from an indented to run-in format if needed, and format entries in any typestyle needed. Understanding what happens with these records in their original state, and then following what happens to them during the compile to online, is the key to successful single-sourcing.

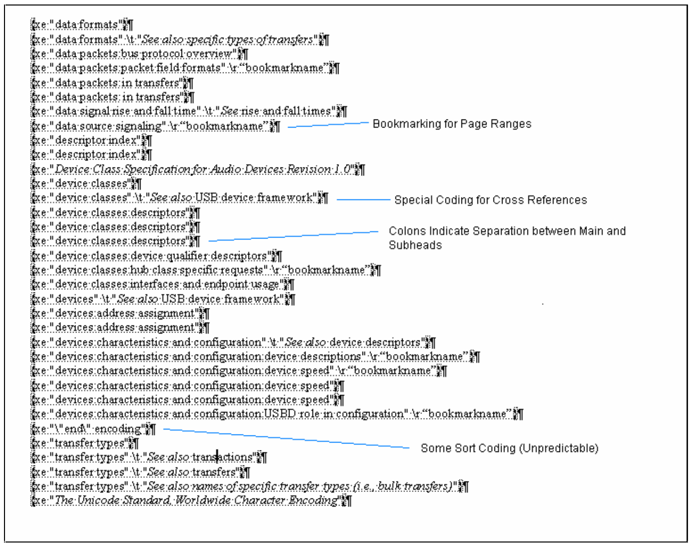
1.2 Embedded Entries in Files

When technical writers use tools such as Microsoft Word or Adobe Frame to create content, and then use the built-in indexing modules to build indexes for that content, the concept of the database is not as easy to see or understand. This is mostly due to the lack of an overall editing view in these programs – all you can see while indexing is the paragraph you are in, or the end compiled result (unless you use a tool like IXGen to build a Frame table for your index editing).

Each XE entry in Word, or each index marker in Frame, needs to be considered as an individual record, and each still needs to have each field in the record understood if the file will be converted correctly to another format. A common feature in embedded indexing modules is the use of punctuation marks to separate main heads from subheads. In Word and Frame, a colon is used. In Frame, a semicolon separates one entry from the next, since Frame allows multiple entries in a marker.

In Figure 3 the same records needed to create the compiled index shown in Figure 1 are displayed as Word XE tags. In real life, these entries would be scattered through the files in the appropriate locations, so that the compiler could pick up the correct page numbers when compiling this index.

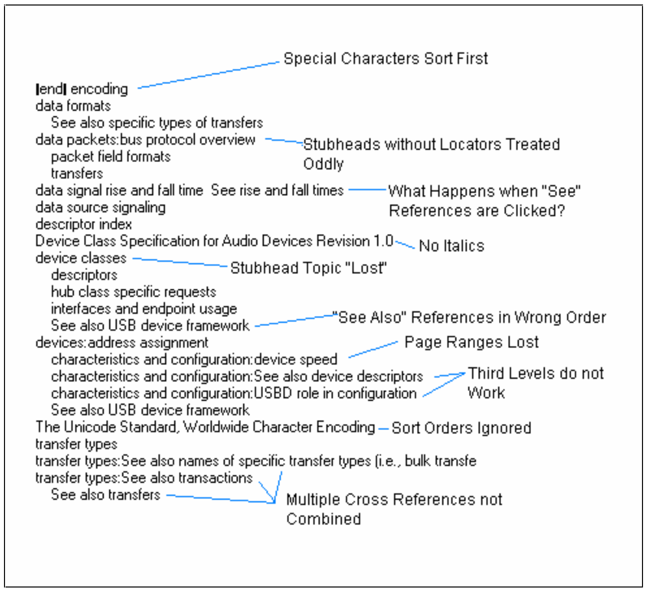
Notice how Microsoft Word loads up each entry with more coding then those recorded in the plain database in Figure 2. When single-sourcing, this coding will have to be understood and dealt with correctly to get each piece of the index to display correctly both in print and in the alternate output. If the codes are not handled correctly in the conversion to an online format, or the indexing is not set up to work in both interfaces, problems result, as shown in Figure 4:



```
{xe "data formats"}¶
{xe "data formats" \t "See also specific types of transfers"}¶
{xe "data packets:bus protocol overview"}¶
{xe "data packets:packet field formats" \r "bookmarkname"}¶
{xe "data packets:in transfers"}¶
{xe "data packets: in transfers"}¶
{xe "data signal rise and fall time" \t "See rise and fall times"}¶
{xe "data source signaling" \r "bookmarkname"}¶
{xe "descriptor index"}¶
{xe "descriptor index"}¶
{xe "Device Class Specification for Audio Devices Revision 1.0"}¶
{xe "device classes"}¶
{xe "device classes" \t "See also USB device framework"}¶
{xe "device classes:descriptors"}¶
{xe "device classes:descriptors"}¶
{xe "device classes:descriptors"}¶
{xe "device classes:device qualifier descriptors"}¶
{xe "device classes:hub class specific requests" \r "bookmarkname"}¶
{xe "device classes:interfaces and endpoint usage"}¶
{xe "devices" \t "See also USB device framework"}¶
{xe "devices:address assignment"}¶
{xe "devices:address assignment"}¶
{xe "devices:characteristics and configuration" \t "See also device descriptors"}¶
{xe "devices:characteristics and configuration:device descriptions" \r "bookmarkname"}¶
{xe "devices:characteristics and configuration:device speed" \r "bookmarkname"}¶
{xe "devices:characteristics and configuration:device speed"}¶
{xe "devices:characteristics and configuration:device speed"}¶
{xe "devices:characteristics and configuration:USBD role in configuration" \r "bookmarkname"}¶
{xe "\"end\" encoding"}¶
{xe "transfer types"}¶
{xe "transfer types" \t "See also transactions"}¶
{xe "transfer types" \t "See also transfers"}¶
{xe "transfer types" \t "See also names of specific transfer types (i.e., bulk transfers)"}¶
{xe "The Unicode Standard, Worldwide Character Encoding"}¶
```

Bookmarking for Page Ranges
Special Coding for Cross References
Colons Indicate Separation between Main and Subheads
Some Sort Coding (Unpredictable)

Figure 3: Entries in Microsoft Word



```
|end| encoding
data formats
   See also specific types of transfers
data packets:bus protocol overview
   packet field formats
   transfers
data signal rise and fall time  See rise and fall times
data source signaling
descriptor index
Device Class Specification for Audio Devices Revision 1.0
device classes
   descriptors
   hub class specific requests
   interfaces and endpoint usage
   See also USB device framework
devices:address assignment
   characteristics and configuration:device speed
   characteristics and configuration:See also device descriptors
   characteristics and configuration:USBD role in configuration
   See also USB device framework
The Unicode Standard, Worldwide Character Encoding
transfer types
transfer types:See also names of specific transfer types (i.e., bulk transfe
transfer types:See also transactions
   See also transfers
```

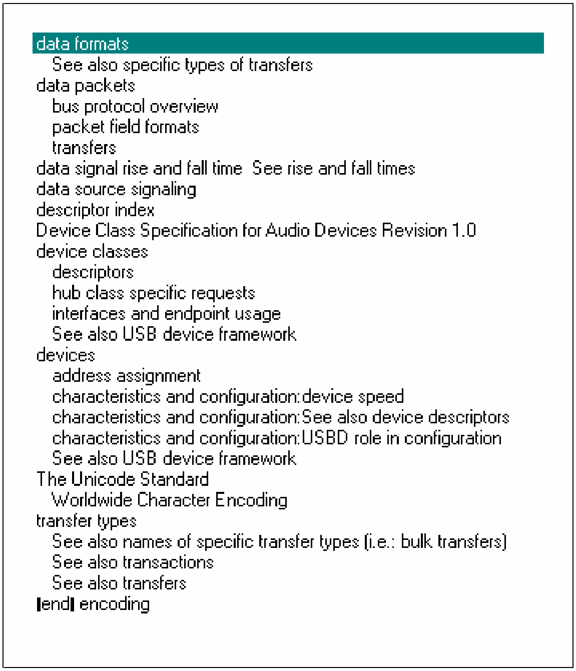
Special Characters Sort First
Stubheads without Locators Treated Oddly
What Happens when "See" References are Clicked?
No Italics
Stubhead Topic "Lost"
"See Also" References in Wrong Order
Page Ranges Lost
Third Levels do not Work
Sort Orders Ignored
Multiple Cross References not Combined

Figure 4: Online Index Entries with Errors

Figure 4 illustrates some of the issues of designing an index for one interface (print) and ignoring the second interface (online).

- Third level headings did not convert correctly for this compiler since it doesn't allow third levels (WinHelp).
- The "See Also" cross references need to sort in a consistent position. Unless the compiler knows to put them at the top of the list of subs or at the bottom, they will sort under "S," in an odd position that makes no sense to the user. And they must sort consistently in every translated language as well.
- The cross references should help the user to get to the preferred terms. Preferably the compiler should be programmed to code these to jump to the "Seen" reference. Otherwise, where do they lead if clicked?
- The compiler doesn't know to combine multiple cross references for online formats.

- Topics that cover multiple pages in print may be broken into several online topics by the author to make better use of the screen interface. The compiler making the index may not pick this up, leaving the online index with only one entry for the first topic, and none for the newly created ones. That leaves a user hanging in the void unless good internal inter-topic navigation exists.
- One large issue is the loss of access to the “device classes” topic that was on page 89. It is now represented solely by the stubhead: if the user fails to click on the stubhead to access it, will the user ever find it? Stubheads raise a usability problem not addressed in print indexes – where should they lead? Should they be active at all? Will users click on them? Take a look at “data formats.” That entry had locators with information. In this interface, will a user know that there is information behind that stubhead?
- No special formatting is applied by this compiler. So all meanings attached to italics or any other special formatting by the user are lost. All indications of the special meaning of cross referencing rely totally on the user to understand the words “See” and “See also”.
- This compiler also ignores any sort codings, arranging the entries strictly by ASCII order, which may not be the best for comprehension. This is best shown by the sorting of the “The Unicode Standard” under “T”.



```
data formats
   See also specific types of transfers
data packets
   bus protocol overview
   packet field formats
   transfers
data signal rise and fall time  See rise and fall times
data source signaling
descriptor index
Device Class Specification for Audio Devices Revision 1.0
device classes
   descriptors
   hub class specific requests
   interfaces and endpoint usage
   See also USB device framework
devices
   address assignment
   characteristics and configuration:device speed
   characteristics and configuration:See also device descriptors
   characteristics and configuration:USBD role in configuration
   See also USB device framework
The Unicode Standard
   Worldwide Character Encoding
transfer types
   See also names of specific transfer types (i.e.: bulk transfers)
   See also transactions
   See also transfers
|end| encoding
```

Figure 5: HTML Help Conversion

Figure 5 shows the same set of entries converted into an HTML Help index. Additional problems crop up in this different compiler:

- Entries using internal punctuation break apart, as in the “Unicode Standard” entry at the end.
- Special characters sort in a completely different order, and are also replaced with the wrong character, as shown in the “end encoding” example.

As you can see with these examples, the single-sourcing process creates problems for index records, and requires the interfaces and the compiler’s behavior to be understood in advance in order to create good indexes in all output formats. Designing the perfect set of entries for all possible output formats isn’t feasible – a lot of special compiling code would have to be written and debugged, and most writing projects do not have the time, staff, or budget to spend creating that coding. Off-the-shelf writing software and tools as well as standard output formats such as HTML Help or PDF are what’s needed by most writing teams, and are the most used tools for accomplishing single-sourcing.

A compromise is nearly always made in the indexes of single-sourced projects. Choosing the best compromise in terms of usability is the difficult task. It can be made easier by running sample small indexes containing all required elements through the process, and then making decisions about what features stay in, and what will not work. Only after testing can the index be designed and written to take advantage of all the interfaces it is required to work for.

2. IDENTIFYING INDEX INTERFACE ISSUES

Every single-sourcing project is using different tool sets, different conversion methods, and different compilers. Tools are chosen to meet a variety of output needs, and sadly sometimes the index isn’t considered when the choices are made. But a long history of successful projects has shown that usable indexes can be developed within any single-sourcing tool set, as long as enough time is granted to do testing, and the indexer is aware of the interface issues. This means the tools for building the content and the final interfaces for presenting the index must be identified early on, and the usability issues for the indexing in the interfaces must be explored as well. Indexing without a solid interface leads to problems.

Some of the questions that need to have answers are:

- How many levels of index will appear? Many tools allow only one level to appear, some allow two. The indexer will have to come up with workarounds in a one-level index, such as using a semicolon to visually divide entries into two levels:

  device classes
  device classes: descriptors
  device classes: hub class specific requests
- When the user clicks on an entry, do they get a list of topics that have that term applied, or is there only a one-to-one correspondence (an entry leading to only one topic) between index entry and topic?
- Are there any controls over the sorting of the entries? Often, the answer is "no" in compiled indexes or embedded indexes converted from print files. The indexer needs to know in order to rewrite problematic entries.
- How do cross-references appear, what types are available, and if the user clicks on them, what happens? This will impact how much double-posting is used in the index design.
- How will the user get to different sections of the index, by clicking a button or typing a letter? How the user navigates in the index is a concern: if the user types in plural words, and you

have indexed with singular entries, they may scroll past the terms they need and never find them.

With all of this in mind, let's embark on an examination of the most common output formats and their index interfaces, to get an overview of where the potential design problems lie.

2.1 Print indexes

Figure 1 showed the diversity of entries and features available in print indexing. Because of its static nature, print indexing allows the most flexibility and creativity in designing the index. A wide range of formatting and punctuation options can be used in a print environment. When single-sourcing, it is usually the print index that is compromised and forced to adapt to the limitations of the other output formats. It is also the easiest to force into other structures.

In the next sections we will show how redesigning the index for the more limited output formats affects the print index as well.

2.2 PDF indexes

Converting content with an index to PDF format is one of the simplest conversions. The final output looks just like the printed piece. There are, however, a few issues to be aware of in designing the piece.

The PDF format usually starts pagination with page one, regardless of whether the piece itself uses a different internal numbering system, such as preface numbers i, ii, etc. If the index is to be interactive and actively link the user back to the desired page, it is best to make the PDF's assigned numbers match the page numbers within the piece. Generating an interactive index is easily done within programs such as Adobe PageMaker or Frame, and as long as the numbering systems match, the user can click the index locators and go to the desired page.

If another tool is used to build the document, the index may not be interactive. Sonar Activate is a tool that activates dead (unlinked) PDF file indexes, and should be considered in these cases, as interactivity increases the index's usability.

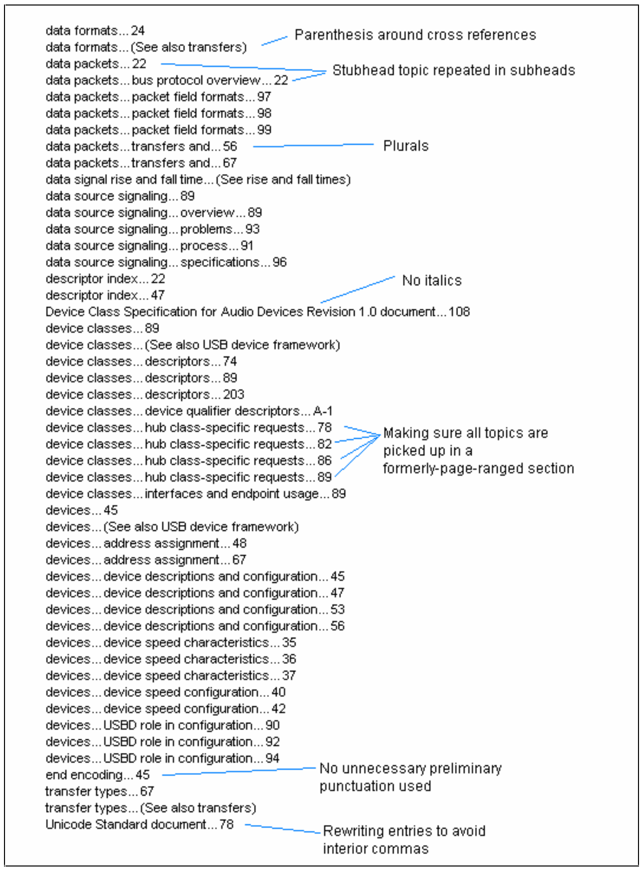
The PDF interface is a bit more difficult to use than a book. It looks the same, but it is harder to flip back and forth without losing one's place. Adding hyperlinks to index entries and tables of contents reduces the user's pain.

Also bear in mind that page-length screens of index entries are harder to browse in PDF format. Scrolling up and down the screen equivalent of a two-column 8 ½ by 11 inch page can be annoying and cause the user to lose track of their task. Shorter pieces with short indexes will work well, but a book with a 50-page index may be too irritating to use.

Cross references in PDF files are not interactive unless the builder chooses to use Acrobat Exchange to hand-link the references to the desired target. This is a usability plus, but is rarely done.

2.3 WinHelp and HTML Help indexes

We have already briefly examined the problems that the WinHelp compiler creates when single-sourcing Word file indexing into online help. Some simple design rules can help make the online index more usable, while keeping the print index relatively usable as well. Once testing is done, developing a style sheet of rules helps to keep compiling mistakes at a minimum. A look at Figure 6 shows the redesign.


Parenthesis around cross references
Stubhead topic repeated in subheads
Plurals
No italics
Making sure all topics are picked up in a formerly-page-ranged section
No unnecessary preliminary punctuation used
Rewriting entries to avoid interior commas

data formats...24
data formats...(See also transfers)
data packets...22
data packets...bus protocol overview...22
data packets...packet field formats...97
data packets...packet field formats...98
data packets...packet field formats...99
data packets...transfers and...56
data packets...transfers and...67
data signal rise and fall time...(See rise and fall times)
data source signaling...89
data source signaling...overview...89
data source signaling...problems...93
data source signaling...process...91
data source signaling...specifications...96
descriptor index...22
descriptor index...47
Device Class Specification for Audio Devices Revision 1.0 document...108
device classes...89
device classes...(See also USB device framework)
device classes...descriptors...74
device classes...descriptors...89
device classes...descriptors...203
device classes...device qualifier descriptors...A-1
device classes...hub class-specific requests...78
device classes...hub class-specific requests...82
device classes...hub class-specific requests...86
device classes...hub class-specific requests...89
device classes...interfaces and endpoint usage...89
devices...45
devices...(See also USB device framework)
devices...address assignment...48
devices...address assignment...67
devices...device descriptions and configuration...45
devices...device descriptions and configuration...47
devices...device descriptions and configuration...53
devices...device descriptions and configuration...56
devices...device speed characteristics...35
devices...device speed characteristics...36
devices...device speed characteristics...37
devices...device speed configuration...40
devices...device speed configuration...42
devices...USBD role in configuration...90
devices...USBD role in configuration...92
devices...USBD role in configuration...94
end encoding...45
transfer types...67
transfer types...(See also transfers)
Unicode Standard document...78

Figure 6: Rewritten Entries for Single-Sourcing in Word Processor or Page Layout Programs and WinHelp/HTML Help

The first redesign is to make sure online topics are not lost under stubheads, and that page-ranged topics, when broken apart online, retain indexing entries. The indexer must be sure to recognize where the topics will be broken up online, and must add entries to each topic that was part of the page range. Stubhead entries are double-posted with a unique subhead, so that they are findable online.

A second redesign focuses on the cross references: in order to keep the "See also" references in a consistent location, and make the "See" references match and be more visible, parentheses are added to the index. Parentheses force the sort order of these special entries to the top of the list of subheads, and also work in every language.

In WinHelp, generic cross references, used to refer the reader to a variety of similar headings without listing them, can still be used, so they can remain in the single-sourced indexing. But in HTML Help, since cross references are active and the target must match the wording exactly, they cannot be used. In either case, all "See also" references must be limited to just one entry – no multiple cross references can be used

Since entries and subentries are sorted in strict ASCII order, there is no way to force the sort to ignore prepositions in subentries. Therefore prepositions must be dropped, and the subentries rewritten.

Internal punctuation causes problems, such as the "end encoding" entry. All such punctuation must be avoided unless the entry is about a special character. The leading underscore (_) underlining commonly found in programming documents can be used if you don't mind seeing a long list of symbol entries at the top of the index. If you are using HTML Help, there are ways to force the sorting, but forcing the sort carries other format restrictions that may interfere with your project. (Check the documentation about the two ways of compiling the project.) Internal commas within entries can also make a compiler break entries apart in unwanted ways, so these are eliminated.

Third level heads do not transfer to online formats, so they must be abandoned for print as well. More detail in the second level heads, and relying on the "Topics Found" box will help rework these entries.

Specialized locators do not translate into online formats, but with embedded indexing, they should still compile correctly, the only loss being the special information provided by seeing the locator.

All italics, bold, or small caps use should be avoided, as they do not translate into online.

Plural entries should be used. If an index uses all singular entries, such as "dog", and the user types in a plural term, such as "dogs", they may go past the helpful entries, as seen below:

```
Dog
  food
  vets
  water
Dogmatic thinking
```

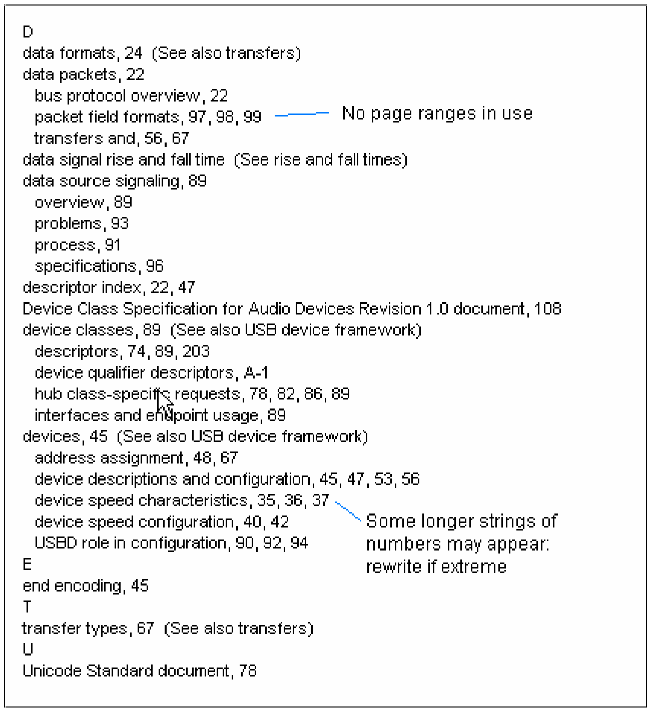
Typing "Dogs" in this case brings the closest entry to the top of the display – "Dogmatic thinking." The user may think there are no entries for "Dogs," as the screen usually rolls the found entry to the top, hiding entries above. Unless the user thinks to scroll up, they may assume there are no entries for their subject.

Tools such as RoboHELP can help the indexer avoid creating dead or "passive" stubheads. In fact, a dead stubhead, one without any topics assigned to it, will be removed during a RoboHELP compile, causing odd-looking subhead compilation. If the index is not built within RoboHELP, these problems will not be evident until the compile. There are several ways to avoid the dead stubhead problem:

1. Assign it to all topics contained in the subheads below it.
2. Assign it to just one "overview" style topic, and make sure that access to the topic also occurs in the subheads.

There's no guarantee that users will click on a stubhead, so access must be duplicated elsewhere.

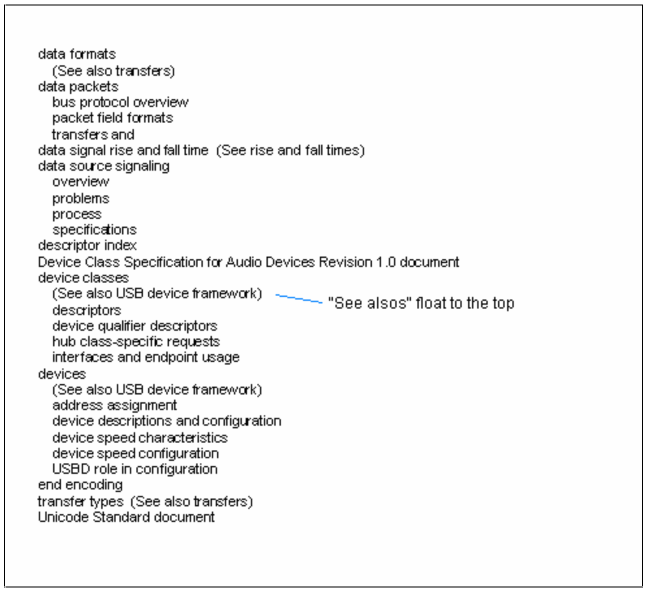
Keeping all of these rules in mind, here are the redesigned entries for both print and online:



```
D
data formats, 24  (See also transfers)
data packets, 22
  bus protocol overview, 22
  packet field formats, 97, 98, 99
  transfers and, 56, 67
data signal rise and fall time  (See rise and fall times)
data source signaling, 89
  overview, 89
  problems, 93
  process, 91
  specifications, 96
descriptor index, 22, 47
Device Class Specification for Audio Devices Revision 1.0 document, 108
device classes, 89  (See also USB device framework)
  descriptors, 74, 89, 203
  device qualifier descriptors, A-1
  hub class-specific requests, 78, 82, 86, 89
  interfaces and endpoint usage, 89
devices, 45  (See also USB device framework)
  address assignment, 48, 67
  device descriptions and configuration, 45, 47, 53, 56
  device speed characteristics, 35, 36, 37
  device speed configuration, 40, 42
  USBD role in configuration, 90, 92, 94
E
end encoding, 45
T
transfer types, 67  (See also transfers)
U
Unicode Standard document, 78
```

No page ranges in use
Some longer strings of numbers may appear: rewrite if extreme

Figure 7: Compiled Index for Print



```
data formats
  (See also transfers)
data packets
  bus protocol overview
  packet field formats
  transfers and
data signal rise and fall time  (See rise and fall times)
data source signaling
  overview
  problems
  process
  specifications
descriptor index
Device Class Specification for Audio Devices Revision 1.0 document
device classes
  (See also USB device framework)
  descriptors
  device qualifier descriptors
  hub class-specific requests
  interfaces and endpoint usage
devices
  (See also USB device framework)
  address assignment
  device descriptions and configuration
  device speed characteristics
  device speed configuration
  USBD role in configuration
end encoding
transfer types  (See also transfers)
Unicode Standard document
```

Figure 8: Compiled Index in WinHelp/HTML Help

As you can see in these examples, the print index makes all the compromises – no page ranges, and extra details to make the online index actually work well. The compromises are not very noticeable to the user. The indexer may need to add more subheads to keep the strings of locators to a manageable length of four or five. This redesign also helps the online index, in that it shortens the lists of topics the user must wade through on screen.

2.4 HTML indexes

Single-sourcing for print and plain HTML indexes is one of the hardest design efforts for the indexer. If truly vanilla HTML is being used, in other words, no special add-ons are developed for the HTML display, the indexer is limited to what is known as a “one-to-one” index. Most other index interfaces, both print and online, are examples of “one-to-many” indexes, allowing multiple locators for each topic, and are easier to build.

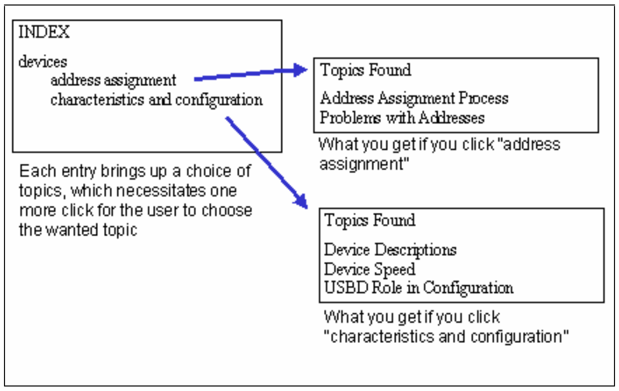

INDEX
devices
address assignment
characteristics and configuration
Topics Found
Address Assignment Process
Problems with Addresses
What you get if you click "address assignment"
Each entry brings up a choice of topics, which necessitates one more click for the user to choose the wanted topic
Topics Found
Device Descriptions
Device Speed
USBD Role in Configuration
What you get if you click "characteristics and configuration"

Figure 9: One-To-Many Indexing

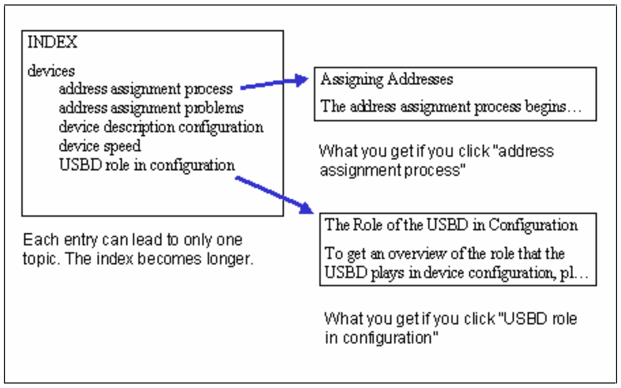

INDEX
devices
address assignment process
address assignment problems
device description configuration
device speed
USBD role in configuration
Assigning Addresses
The address assignment process begins…
What you get if you click "address assignment process"
Each entry can lead to only one topic. The index becomes longer.
The Role of the USBD in Configuration
To get an overview of the role that the USBD plays in device configuration, pl…
What you get if you click "USBD role in configuration"

Figure 10: One-To-One Indexing

In a “one-to-one” index, each entry in the index can lead to one location and one location only. Unlike the string of page numbers in many print indexes, or the set of topics in a “Topics Found” dialog box in WinHelp, you can only present the user with one topic. HTML indexes are notorious for this – unless enhanced by programming, each entry in the index can lead to only one anchor mark. Indexers have tried various methods of presenting the index to overcome this limitation, but these experiments have not been very successful. For example, the American Society of Indexers web site index tried the following format:

Tutorials 1, 2, 3, 4

Without subheads, or clues to the content of each number, each link had to be clicked on separately, the page examined, and the user has to then navigate back to the index to try another link. ASI has since abandoned this design in favor of a simpler “one-to-one” system. The disadvantages of the one-to-one system are that more subheads are required to create unique pathways to each link. The indexes become much longer. Long indexes online are hard for users: scrolling and flipping through pages is frustrating.

An alternate creative design concept would be to bring up a separate page listing third level heads or topic titles to help the user. The coding required to produce this effect automatically would probably prohibit this concept, but if the pages were to be updated and generated frequently, and an automated way of generating the pages devised, it might be worthwhile:

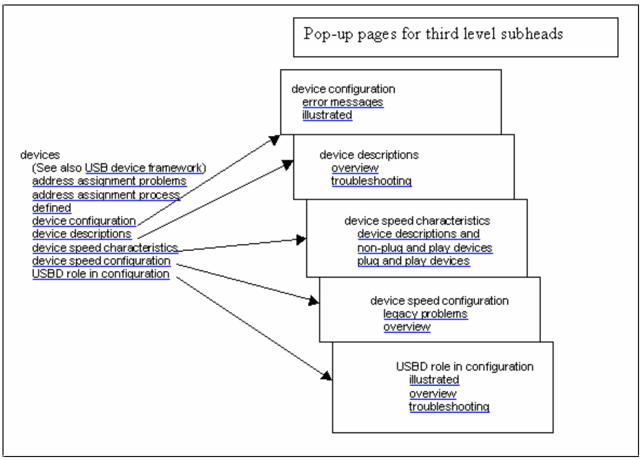

Pop-up pages for third level subheads
devices
(See also USB device framework)
address assignment problems
address assignment process
defined
device configuration
device descriptions
device speed characteristics
device speed configuration
USBD role in configuration
device configuration
error messages
illustrated
device descriptions
overview
troubleshooting
device speed characteristics
device descriptions and
non-plug and play devices
plug and play devices
device speed configuration
legacy problems
overview
USBD role in configuration
illustrated
overview
troubleshooting

Figure 11: Adding Pop-up Pages for HTML Index Details

Below are index entries rewritten for HTML single-sourcing.

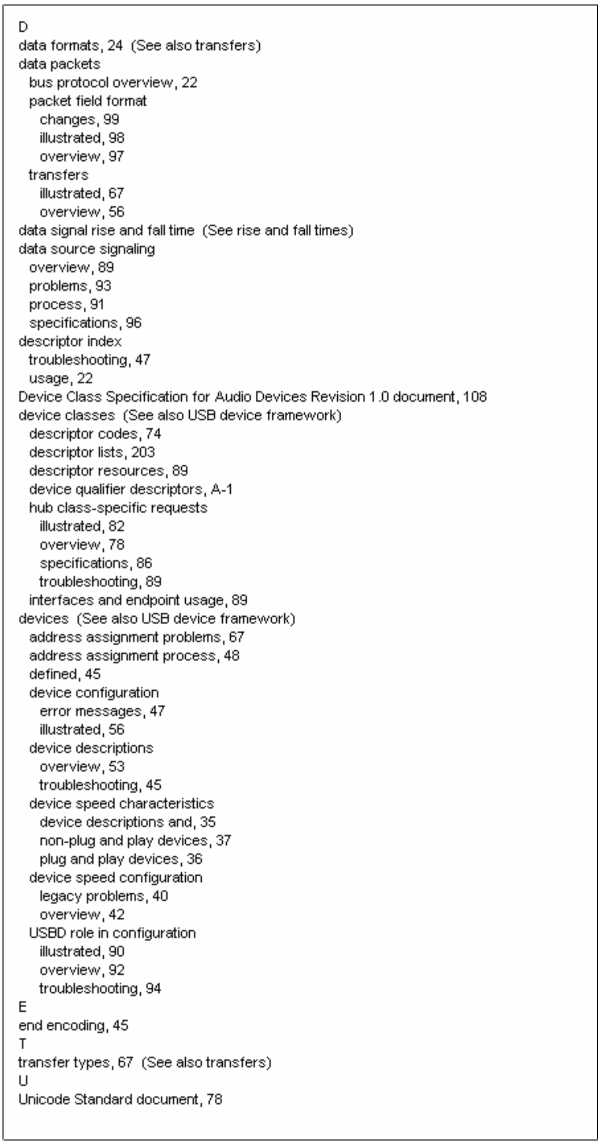
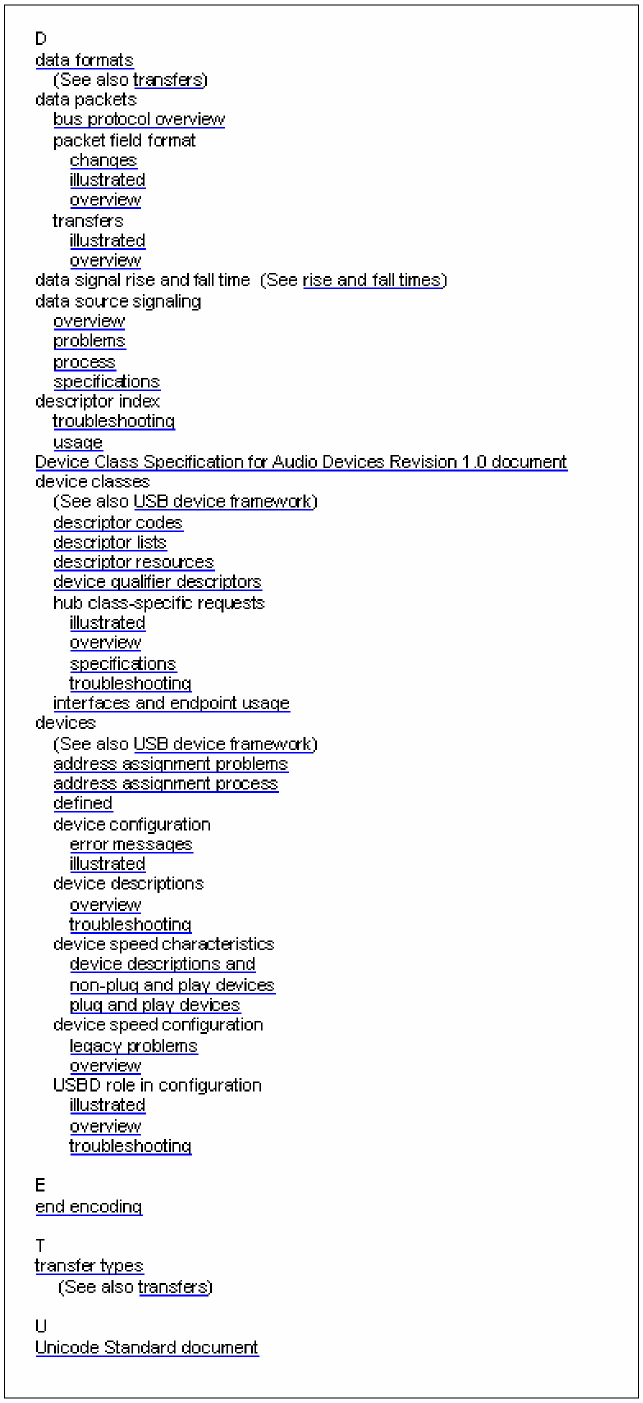

D
data formats, 24 (See also transfers)
data packets
  bus protocol overview, 22
  packet field format
    changes, 99
    illustrated, 98
    overview, 97
  transfers
    illustrated, 67
    overview, 56
data signal rise and fall time (See rise and fall times)
data source signaling
  overview, 89
  problems, 93
  process, 91
  specifications, 96
descriptor index
  troubleshooting, 47
  usage, 22
Device Class Specification for Audio Devices Revision 1.0 document, 108
device classes (See also USB device framework)
  descriptor codes, 74
  descriptor lists, 203
  descriptor resources, 89
  device qualifier descriptors, A-1
  hub class-specific requests
    illustrated, 82
    overview, 78
    specifications, 86
    troubleshooting, 89
  interfaces and endpoint usage, 89
devices (See also USB device framework)
  address assignment problems, 67
  address assignment process, 48
  defined, 45
  device configuration
    error messages, 47
    illustrated, 56
  device descriptions
    overview, 53
    troubleshooting, 45
  device speed characteristics
    device descriptions and, 35
    non-plug and play devices, 37
    plug and play devices, 36
  device speed configuration
    legacy problems, 40
    overview, 42
  USBD role in configuration
    illustrated, 90
    overview, 92
    troubleshooting, 94
E
end encoding, 45
T
transfer types, 67 (See also transfers)
U
Unicode Standard document, 78

Figure 12: Print version of HTML Single-Sourcing File

Both the print index and the HTML index appear overly long, overly detailed, but still usable. With raw HTML, it's my personal opinion that the stubheads should be passive or dead, to remove complexity from the screen presentation. Active links are underlined, and clear to the user.



```
D
data formats
   (See also transfers)
data packets
   bus protocol overview
   packet field format
      changes
      illustrated
      overview
   transfers
      illustrated
      overview
data signal rise and fall time  (See rise and fall times)
data source signaling
   overview
   problems
   process
   specifications
descriptor index
   troubleshooting
   usage
Device Class Specification for Audio Devices Revision 1.0 document
device classes
   (See also USB device framework)
   descriptor codes
   descriptor lists
   descriptor resources
   device qualifier descriptors
   hub class-specific requests
      illustrated
      overview
      specifications
      troubleshooting
   interfaces and endpoint usage
devices
   (See also USB device framework)
   address assignment problems
   address assignment process
   defined
   device configuration
      error messages
      illustrated
   device descriptions
      overview
      troubleshooting
   device speed characteristics
      device descriptions and
      non-plug and play devices
      plug and play devices
   device speed configuration
      legacy problems
      overview
   USBD role in configuration
      illustrated
      overview
      troubleshooting

E
end encoding

T
transfer types
   (See also transfers)

U
Unicode Standard document
```

Figure 13: Vanilla HTML Index

2.5 Conditional Text Single-Sourcing

Index design becomes increasingly complex if the single-sourcing project also includes analysis of conditional text sections or DHTML pop-up text. Some projects have the files split by macros and recombined into various forms for print, for international versions, or for light and professional versions. The most extreme case of this in my experience was a project involving a basic user manual that was created for both print and online, with some text appearing solely in the print version, and other portions appearing solely online. This would have been easy, except the base book was then combined with three other specialized print and online manuals, to generate four separate products. The single-sourced base book and base online help appeared in all four products, and then was augmented by additional print/online files, and online-only files geared toward the specific product version. A total of 22,000 index entries were needed to cover the entire product range. If a user bought multiple products, the indexes from all the products merged in the online help screen. All 22,000 entries had to work whether displayed for just one product, two, three, or all four.

As an extra added bonus, portions of the files had to be done early, frozen, and sent for translation. To an indexer, this can be a nightmare, especially in terms of creating stubheads that do not get lost when merged with other subheaded topics.

Conditional text scenarios like this one can be handled in two ways: index everything and match the terminology across the products as much as possible, but pay no attention to how things look when conditional entries drop out of the merged index. Or guarantee that the indexes will work and be usable across the board, regardless of which index entries are combined and displayed to the user.

The second approach is more costly and takes longer, but in the end serves the user better. I was lucky that the company chose the second method for several of its releases, and the documentation team experimented with a variety of methods to make the indexing, the single-sourcing, and the translation efforts to generate twenty-two language versions go as smoothly as possible.

In various releases, the following tool sets were used:

- Base files:Word files → Final files: PageMaker and WinHelp (RoboHELP)
- Base files:Word files → Final files: PageMaker and HTML Help (customized in-house tools)
- Base files:Word files → Final files: FrameMaker and HTML Help with DHTML (customized in-house tools)
- Base files:Word files → Final files: FrameMaker and vanilla HTML output (customized in-house tools)

If faced with a task this size and complex, testing the outputs is essential before beginning. Once the output files are tested, and the compromises decided upon, indexing can begin.

It's best when faced with this much complexity to index outside the files – Excel, with its flexible columns and color coding allows the indexer to be able to sort and check each product's standalone indexing for problems, and sort and check each product's indexing against the whole series before inserting final entries into the files. The index spreadsheet tracks the filename, topic name, product, entry text, location, and special notes. (See Table 1) As yearly revisions are worked on, old indexing already known to be tried and true can be reused – updated with new filenames and topic numbers as needed and incorporated.

Product	Filename	Print or Online	Topic ID	Entry	Notes
Basic 4.0	Intro	both	Saving_your_files	saving:files	
Basic 4.0	Intro	both	Saving_your_files	files:saving	
Basic 4.0	Intro	Online	Saving_your_files	web pages:saving as	Leads to menu of choices
Professional 4.0	Newfeature	both	What_s_new	Professional:new features	

Table 1: Conditional Text Indexing Spreadsheet

Only certain columns are be stripped out and used from this spreadsheet. Macros can be built in-house to place entries automatically at the beginnings of the topics according to the recorded topic IDs. Macros can also used to generate the list of topic IDs to be indexed, minimizing typing mistakes. All of the customized macro sets need to be tested with sample entries so that the outputs could be predicted before indexing begins.

Conditional text single-sourcing leads to compromises. When portions of content and entries are removed, you can get index entries that do not make sense alone. Or if portions of content and index entries are added, you can get stubhead entries that really needed a subhead to make them findable. The budget will dictate how much compromise can be tolerated. Designing the index for the online portion makes the most sense in these cases, as it is usually far less flexible than the print index and cannot accommodate the niceties of print.

DHTML adds more flavor – if the text is not going to be visible on first glance to the user, should it be indexed? It may appear on the page in print, and so should be indexed, but if it is buried under a drop-down or pop-up listing, it most likely should not be indexed. Again, the print portion loses in this compromise.

2.6 XML Indexes and Custom Index Interfaces

With the entry of XML into the documentation world, and associated content management systems, the development of single-sourced indexes could become much easier. XML content databases may be able to get around the restrictions of off-the-shelf software packages. As more and more groups use these tools, and as they become more affordable, we may see better single-sourced indexes. But that time is not here yet. And when it arrives, the index interfaces generated by the tools still need to be tested to be sure that records are entered in the most usable and retrievable manner.

The hardest work will be the upfront design of the indexing portions of the DTD, and determining if one set of entries can be manipulated for each output style. There is the potential for two or three sets of index entries (one for each output style) to be included for chunks of contents, if that turns out to be the best method. One of the biggest benefits, however, will be allowing the index content to remain free of external coding requirements.

Analysis of the output interfaces must be done before developing the DTD sections or adding entries to the content. In many cases, the output will be going into a customized index interface, and if it is possible to have an indexer involved with the design of the interface for that output, all the better.

When working with customized index interfaces, make use of Figure 1, which displayed all the kinds of index components that can be used in technical indexing. Usability features that could be incorporated into a customized interface are:

- Type Ahead Capability
- Interactive Cross References
- Topics Found Mechanism
- Stay on Top Functionality
- Search (within the index to bring up entries with plural endings, gerund endings, or related terms if the term doesn't exist)

If dealing with a customized index interface that is set in stone and cannot be changed, devise a testing plan to make sure all the files will work when run through the compilation.

3. TESTING PLANS

Such a variety of different tool sets are being used to single-source documentation that no one tool set recommendation can be made. As pointed out earlier, the budget and needs of the rest of the documentation set dictate the tool choices, and the index must find a means of coping with the choices.

Doing preliminary test runs of documents with sample indexing through the entire tool set should allow the indexer to predict what will happen in each interface before adding entries to the real content. A sample index incorporating all wanted elements should be developed. For ideas and needed elements, use the entries from Figure 1, as it contains most elements present in complete technical indexes. Then eliminate what does not work, and create a set of style suggestions to make the index work in all the necessary formats.

Any specialized macros built to assist in the project will need to be set up early on so that special needs for punctuation (dividers between entries, or separators between main heads and subheads) can be tested at the same time.

4. CONCLUSION

As these examples have shown, designing indexes that work equally well in any of the various interfaces in which a document may be displayed, whether print, help, PDF, or on the Web, presents challenges. Testing the tools for problems before starting to index is the only way to ensure a usable index in all formats. Once testing is done, a basic style sheet can be employed by all involved in the indexing effort to keep entries consistent and workable throughout the conversions.

5. ISSUES CHECKLIST

Below is a short list of problems to check for in the design process and conversion tests.

Structure:

- Levels available – one, two, three?
- One-to-one or one-to-many interface (multiple locators)
- “Topics Found” mechanism present?
- Multiple entry sets available (XML)?
- Pop-up pages available?

Cross references:

- Target names match perfectly or not?
- Sorting issues
- Interactivity
- Limited to one reference or multiple?
- Generic references available

Punctuation:

- Separator punctuation within entries
- Divider punctuation between lists of entries
- Internal punctuation within entries

Formatting

- Character formats such as italics, bold, or small caps available?
- Display of special characters
- Plural entries

Sorting

- Sorting of prepositions and lead stop words
- Sorting of special characters

Stubheads

- Active or passive stubheads
- Double posting under stubheads

Other issues:

- Internal numbering systems in PDF display
- Conditional text handling
- Lost online entries from print page ranges
- Long strings of locators in print index

6. ACKNOWLEDGMENTS

My thanks to Microsoft Corporation and the former Visio Corporation for providing indexers with single-sourcing projects in nearly every index format imaginable. My thanks also to Caroline Parks for sharing the challenges of single-sourcing indexing through many projects and tools, and for her comments on this article.