



XML

NISO

dtbook3

Braille2000

Wondering what all that means? Computer people are so proud of their acronyms and letter abbreviations! If you read what follows, you will find out what these symbols mean and how they relate to the process of producing textbook braille. The first hint is that *Braille2000* is a new direct-entry and translation tool for braille for Microsoft Windows, being developed by Computer Application Specialties Company. It is targeted for release in the 2nd quarter of 2001. It will ultimately replace ED-IT PC and it will fully support ABT files.

The first symbol, XML

The story about XML begins about 25 years ago. A person named Charles Goldfarb invented an artificial language for document description called "Standard Generalized Markup Language". Right away everyone was calling it SGML, but for years it remained purely an information technology research curiosity. In the latter '80s, other university experimenters were developing a simpler document language for use on the then-infant "Web". This application of SGML was called the Hypertext Markup Language (HTML), because of the heavy use of colored text that takes you to a different (linked) document when you click on it—linked text is called "hypertext".

As Web-this and Web-that started happening everywhere, additional capabilities were desired. By 1996 work was underway, principally at Microsoft, on an extended kind of HTML, called XML. Jean Paoli and others wrote the XML specification in 1996. Like HTML, XML is a subset of the original SGML. But unlike SGML, XML is designed to be less abstract and less complex—when it comes to Web-based documents, lean and explicit control statements lead to easier to build and more reliable Web documents (i.e., Web-pages). But what does Web-this and Web-that have to do with braille? Good question. But first, a wee bit of additional insight.

Back in the latter '80s (circa Windows 1.0 which few people used because it wasn't very good), word processing meant editing plain-text files on a computer, typically a Unix computer (Unix is an operating system, like Dos and like Windows), at a university. The final copy would be printed on a daisywheel printer, like an automated typewriter. There were no font choices or character size choices, and you were really living if you had a dual-wheel daisywheel printer on

which the second wheel held Greek letters and standard math symbols. To avoid the tedium of rearranging paragraphs when the document was revised, the document files were prepared (still with a simple text editor) with unformatted lines of text mixed with layout directives that described the paragraph margins and indents. These added control statements were "markup" for the prose. It wasn't nearly as complicated as SGML, but it was never the less a very simple markup language. Then everything changed.

The magic moment was the advent of the Apple Computer "laser printer" (Xerox had done it before for business clients, but prices were very very high). Now, using the new laser printer, a Macintosh computer could prepare documents with a variety of fonts and letter sizes. The output had the look of typeset, professional printed materials. It goes without saying that with such a capability, "word processing" now meant managing the fonts, and sizes, and styles of the print as well as paragraph and page layout. The ad hoc markup mechanisms were extended to describe the desired font, font size, and italic or bold attributes, etc. As soon as computer graphics technology became affordable, then the on-screen image of the document was made to look like the final printed copy, and "what you see is what you get" (WYSIWYG) became the standard for document editing. At this point, the user stopped manipulating the markup control elements directly—the WYSIWYG editor did it for him.

Up until nearly 1996, "markup controls" were considered important for controlling the way the document was displayed or printed. I.e., the markup described the "presentation" of the document. This kind of markup can be found internal to Microsoft Word files, to WordPerfect files, and in standard file formats such as the so-called "Rich Text Format". In the latter case, the markup is in the form of added text (as it is in HTML and SGML). In the two former cases, the markup is in the file in a binary (non-text) form—it's there but you can't see it without a special editor. In all of these presentation-based markup languages, the markup symbols are defined by the tool that will present the text on the screen, i.e., by WORD or by WordPerfect.

XML is yet another document markup language. Its control statements can certainly be used to describe the way you want a document to be presented. But by 1996, much more elegant ideas came into mind—that of using markup phrases (which are called "tags") to organize the character data into sections, using the tag to denote the role or purpose for the text. Instead of merely cueing the presentation of the text, tagged text could be used abstractly like a kind of simple database. With the right software, a document or large set of documents could be queried for information by tag, with the text from selected elements (selected by matching tag words) becoming the result of the query. In XML, unlike the presentation forms listed above, the markup tags are defined by the document, not the tool that will present them on the screen. This added abstraction is what separates XML and SGML from the other formats.

And then it occurred to information scientists that a highly abstract document could house text that could be presented in not just one way, but in several alternative ways, once the tags were used to denote abstract document structure rather than just presentation. A set of structure tags can be converted to a different set of presentation tags through the use of a specification called a "stylesheet". This relatively new idea is likely to be golden to the transcription of textbooks into braille. Yes, you are right, how that is so is not yet clear. Read on.

The transcription of textbook material into braille is a time-consuming task. There are roughly four things that are involved: (1) translating the prose into contracted braille prose, (2) arranging

the layout of the prose appropriately for braille-sized pages, (3) constructing the preliminary pages, and (4) providing any special transcription elements using specialized codes such as Nemeth math code or Computer Braille Code, etc. To stay to the point of his commentary, let's assume that items 3 and 4 get done by computer-manual methods. There is still a lot of work bottled up in items 1 and 2, and in doing those things, information about the presentation of the textbook print copy is of very little use. The basis for the layout of the braille pages is, in general, the structure of the textbook. The salient point is that presentation is not structure. Word processor documents typically deal with presentation. And there's the rub—braille page layout algorithms, such as they exist, would have very little to go on. Enter XML and NISO and document structure.

The second symbol, NISO

Since 1999, the National Information Standards Organization (NISO) has been working on defining an abstract structure for digital documents in support of digital talking books. Think of a digital talking book as a computer with a synthetic voice that can read any digital (file) text you give it. Give it digital book text and it begins reading. And if you wanted to read from the first page straight through to the last page, you would be done inventing a digital talking book. But of course the listener would rather be able to navigate through the structure of the book, instead of listening to it only linearly. NISO turned to XML as a way to capture the structure (not presentation, which is irrelevant to a talking book) of a textbook. Information scientists observed that it was better to capture as much of the structure as possible, down to the paragraph level, given that new traversal strategies would be invented. They called their XML specification for the purpose of conveying textbook structure "dtbook3". The "dt" is for digital talking, and the 3 is simply the third revision.

The third symbol, dtbook3

"dtbook3" is an XML specification for a type of XML document file. In XML parlance, dtbook3 is a "document type definition". It states the names of the text tags that a valid document can use and how the tags should be used to convey the structure of the document (a document could be an entire book or a book chapter). NISO's idea is to convince textbook publishers to generate the dtbook3 kind of XML document as an overhead process in support of digital talking book users.

Then someone pointed out that many of the braille page layout issues revolve around document structure (again, structure not presentation). The open question is this: "can structured documents prepared to the dtbook3 XML specification be milked for page layout (and translation cues) for the production of braille textbooks?" It would appear that the answer is "Yes, at least in part". That still leaves room for a wide range of degrees of success. Research in that area is ongoing. But it could mean great gains for braille production, especially if publishers willingly supply dtbook3 XML files, for digital talking books, for large print readers, and for braille production. If it works, one XML file should serve all those needs (but at the current time, that capability has not yet been demonstrated).

The fourth symbol, Braille2000

At the time word of the NISO XML work on dtbook3 came to Computer Application Specialties (spring of 2000), work was already under way on a new replacement tool for ED-IT PC, a tool that would better fit the more modern software environment of Windows2000 and WindowsME. Given that this project began in 1999, the code name for this project was "Braille2000". If development remains on schedule, the goal is to release the first edition of Braille2000 around May of 2001 (and no, it won't be called Braille2001, the trade name is meant to refer to the 2000 millennium in general).

In the past months, the internal mechanisms of Braille2000 have been altered to be XML mechanisms. Braille2000 is being prepared to read dtbook3 XML files and to use the structure-denoting tags to guide the layout of the braille pages. It is hoped that page layout driven from the dtbook3 information will be accurate and reliable. But even if it is imperfect, Braille2000 is still a powerful WYSIWYG braille editor, like ED-IT PC before it. It should be easy to tidy any imperfections of XML-aided translation in the course of the transcriber's work on the unautomated or less automated aspects of the textbook transcription.

The internal strategies of Braille2000 are based on standard XML techniques. Braille2000 directly understands a dialect of XML for describing braille documents. This XML sub-language (defined in a document type definition called "brl2000a") conveys the presentation of the braille material. An XML "stylesheet" (for braille) will be used to convert the dtbook3 XML statements (structural cues) into brl2000a statements (braille page presentation). It is helpful that this conversion is from one XML dialect to another XML dialect (there are standardized tools and approaches for doing this kind of task).

Braille2000 will be able to read and write XML braille documents (although as of this moment, there are no other XML braille tools that could read them). It will also be able to read and write a variety of other kinds of braille file formats, including the ABT format used by ED-IT PC. The native file format of Braille2000 will continue to be the "Annotated Braille Text File" as is used by ED-IT PC, however the new version of that file type will contain XML control statements rather than the ad hoc ED-IT PC markup language that was invented for ED-IT PC. (Yes, the ED-IT PC .ABT kind of file contains an ad hoc markup language within it. In Braille2000 that markup language will be XML. Transcribers did not see the ED-IT PC markup language before, and they won't see the XML variety now.)

Although XML is an important part of Braille2000, few transcribers will need nor want to learn about XML itself. When using Braille2000, the transcriber will not see nor directly manipulate XML statements. It may be that from time to time, a publisher's XML file may arrive flawed, and for that situation, a few transcribers may wish to manipulate the publisher's file. There are commercial tools for doing that, such as Corel WordPerfect Version 9.

Like ED-IT PC, Braille2000 will also be a superb tool for direct-entry braille transcription and for WYSIWYG editing of braille files. It will be another step up in capabilities and will have improved support for Nemeth math transcription.

The XML/dtbook3 translation capabilities will be in Braille2000: The Document Processing Editing, which is targeted for release in the fall of 2001.

