The Annotated Rainbow DTD

Describes Rainbow rev number 1.9
6 December 93

0. Legal notes

Copyright 1993 Electronic Book Technologies, Inc.

Permission to use, copy, modify and distribute the Rainbow DTD and this "annotated DTD" documentation for any purpose and without fee is hereby granted, provided that this copyright notice appears in all copies. If you modify the Rainbow DTD, rename your modified version and make it clear that it is not the "official" Rainbow maintained by EBT.

EBT makes no representation about the suitability of the DTD for any purpose. It is provided "as is" without expressed or implied warranty.

The Rainbow DTD is maintained by EBT, Inc. Please direct all questions, bug reports, or suggestions for changes to "dfs@ebt.com".

Please note that an SGML declaration is provided for this DTD. It appears in the raw DTD but not in this annotated version.

Public Identifier: "-//EBT//DTD Rainbow 1.0//EN"

1. Goals

Rainbow is an SGML document type designed to represent documents that have been created via word processor (WP) technologies (e.g. MSWord, WordPerfect, Frame, Interleaf). Design goals:

1. WP-independent
2. Formatting-based: the mapping from WP to Rainbow should be as simple as possible
3. Unambiguous: there should be only one way (or very few ways) for a given document to be represented in Rainbow
4. Suitable interim step in the passage from WP to useful SGML (i.e. stores information needed in the recognition of content)
5. Simplicity

2. Limitations

Completeness in representation of WP data is not a goal. When one considers the maturity of today's WP technologies (many of which now include significant page-layout and drawing features) and the rate at which that industry continues to mature, one immediately realizes that any attempt towards completeness is doomed to failure.

Thus, we have decided *not* to set such goals as:

- support for recreation of the original source
- native representation of equations
- support for "non-prosaic" documents (e.g. a WordPerfect representation of an employment-application form, with chains of underscores representing form blanks, etc.)
- retention or handling of WP "layout" features (e.g. multi-column)

Our goal is to handle a small, core set of features that are common to all WPs and that are the bread-and-butter of most prosaic documents. We acknowledge that there are conversion projects for which Rainbow is not suitable.

3. Glossary of Key Terms

paragraph, or paralevel object	A WP object that starts/ends with explicit line breaks. My use of this term does not match the use that would be espoused by an English teacher. My "paragraph" might be a section title, or a note, or a list item.
marked paragraph	A paragraph whose content begins with a marker.
marker	An object that "adorns" a paragraph, typically (but not always) having an exaggerated "hanging" indentation. Typical markers are bullets, dashes, numbers, letters.
paratype	Most WPs allow the author to specify that a paragraph is of a given type. In MSWord, each paragraph can be assigned a "style"; that is its paratype.
paralevel formatting attributes	Attributes that can only be assigned to a paragraph as a whole. Abbreviation: PLF Examples: space-before, left-indent, first-indent
charlevel formatting attributes	Attributes assigned to a contiguous range of text inside a paragraph, or assigned to a paragraph as a whole Abbreviation: CLF Examples: font-weight, font-slant
override formatting	Applies only to PLF and CLF formatting for a paragraph that has a paratype. WPs that support paratypes allow the user to specify formatting for the paratype. This formatting definition is thus automatically applied to all paragraphs assigned that paratype. These WPs usually allow the user to "override" these paratype-level formatting attributes, on a per-paragraph basis.

4. The Annotated DTD

```
<!ELEMENT rainbow  - - (FILEINFO?,STYINFO?,DOC)>
```

General information about the document.

```
<!ELEMENT fileinfo - O EMPTY>
<!ATTLIST fileinfo     origin   CDATA #IMPLIED
                       dtdver   CDATA #IMPLIED  -- Version of Rainbow DTD being
                                                   used for this instance -->
```

Third parties implementing "Rainbow Makers" should consult with EBT to register origin names; note that origin names should include the version of the WP (e.g. use "MIF-3" instead of just "MIF"). A multi-WP standard like RTF should be represented by a string that includes identification of the WP that generated it (e.g. "MacWord-RTF-*versionnum*" or "WinWord-RTF-*versionnum*").

```
-------------------------------------------
```

The formatting (PLF and CLF) attributes are present for one reason only: to allow recognition of content. The purpose of these attributes is ***not*** to facilitate recreation of the formatting or layout of the original WP document. Thus, the set of attributes is not complete.

Remember: a paralevel object can have both PLF and CLF attributes; but a subparalevel object can only have CLF attributes.

Here is the DTD specifications for the PLF and CLF attributes. Comments are used to specify the semantics.

```
<!ENTITY % boolean "NUMBER"  -- 1 means yes, 0 means no -->

<!ENTITY % plf-att -- Paragraph Level Formatting --
"left-indent CDATA #IMPLIED
       -- measured in points; absolute number; dont use "+=" notation --
 right-indent CDATA #IMPLIED
       -- measured in points; absolute number; dont use "+=" notation --
 first-indent CDATA #IMPLIED
       -- same as in DynaText: offset from the left-indentation --
 justification (LeftJust|CenterJust|RightJust|FullJust|InJust|OutJust) #IMPLIED
       -- FullJust means both left and right justification
          InJust/OutJust means justify towards binding/perimeter --
 line-spacing CDATA #IMPLIED
       -- same as in DT: distance between baselines --
 space-before CDATA #IMPLIED
 space-after CDATA #IMPLIED
       -- both of the above are as in DT: measured in pts --
 keep-with (Next|Prev) #IMPLIED
       -- keep with (next/prev) paragraph --
 tab-stops CDATA #IMPLIED
       -- colon-separated numbers; each number is a measurement (in points)
          of the tabstops distance from the left-indentation --
 keep-together %boolean; #IMPLIED
       -- does the author demand that the paragraph not cross page boundaries? --
 pg-brk-before %boolean; #IMPLIED       -- page-break before --
 col-brk-before %boolean; #IMPLIED      -- column-break before --
 border NUMBER #IMPLIED
       -- value is sum of the appropriate members of this list:
                1 for top
                2 for right
                4 for bottom
                8 for left   --
>

<!ENTITY % clf-att -- Character Level Formatting --
"charset CDATA #IMPLIED
 font-family CDATA #IMPLIED
 font-size CDATA #IMPLIED
 font-weight (Medium|Bold) #IMPLIED
 font-slant (Roman|Ital)   #IMPLIED    -- Use Ital both italic and oblique --
 score-location (Under|Over|Through) #IMPLIED
 score-type (Single|Double|Dotted) #IMPLIED
       -- score-type is ignored if score-location is not set --
 vertical-offset CDATA 0
       -- as in DT: offset from the baseline, in points, - for sub, + for sup --
 foreground CDATA #IMPLIED
 background CDATA #IMPLIED
       -- If the generator can do so, it must specify color as a RGB
            hexadecimal numbers in this format:  "#rrggbb"
          If the generator is unable to do so, it can use an arbitrary
            representation (e.g. name, index number).
          Note that if the latter representation is used, it will not
            be possible for the Rainbow consumer to reproduce colors
            in its rendering facilities. --
```

```
        -- Note: if the text is not in a "special" color, don't set
                 the foreground attribute. --
        -- Note: if the text is not on a "special" background, don't set
                 the background attribute. --
        -- Example: use #888888 for background in the typical case of "shaded
                    background" for black-on-white printing --
  lowercase-display (SmallCaps|FullCaps)  #IMPLIED
        -- Note that this specifies how lowercase data is displayed;
           this does not describe the data itself. --
  outline %boolean; #IMPLIED
        -- Is each letter being displayed as a hollow outline? --
  change-bar (BarLeft|BarRight) #IMPLIED
  hidden %boolean; #IMPLIED"
>

-------------------------------------------
```

The WP document's stylesheet (if any) must be stored in the STYINFO section.

```
<!ELEMENT styinfo  - - (PARATYPE+)>

<!ELEMENT paratype - O EMPTY>
<!ATTLIST paratype     name     CDATA #REQUIRED
                       %plf-att; %clf-att;>
-------------------------------------------
```

The representation of the document's content. Note the one inclusion exception.

```
<!ELEMENT doc      - - ( HEAD?,
                         (((PARA|ILLUS|TABLE|SYSOBJ)+,
                           STRUCLVL*)
                          |(STRUCLVL+ ) ) )

                       -- inclusion exception --
                       +(NEWPAGE|IDXTERM)          >
-------------------------------------------
```

Many word processors support representation of hierarchy through implication. For example, an MS Word document is essentially a flat list of paralevel objects, each of which can be assigned a paratype. Use of paratypes like "Heading 1", "Heading 2", etc. — applied to paralevel objects that represent section titles — allow this flat list to represent a hierarchy of arbitrary depth.

The implied hierarchy can be represented ***explicitly*** in the Rainbow via the STRUCLVL element. The process that generates Rainbow can watch for the title-bearer paratypes and use them to generate STRUCLVL elements.

If the original document does not contain title-bearer paragraphs, the generated Rainbow document will of course not contain any STRUCLVL elements. In such a case, it is up to the next process (the one that takes in Rainbow) to attempt to recognize title-bearers (by using the formatting information in the Rainbow).

```
<!ELEMENT struclvl - - (HEAD,(PARA|ILLUS|TABLE|SYSOBJ)*,STRUCLVL*)>

-------------------------------------------
```

Here is the most important element of them all. Actually, it's a pair of elements that are virtually twins. The only difference between a HEAD and a PARA is that the former is an object that has been given "title-bearer" status by the author (e.g. by giving it a paratype of "Heading X" in Microsoft Word). Any reference to "paragraph" in this specification is actually a reference to both HEADs and PARAs.

The value of the paratype attribute (if present) must match the name attribute of exactly one PARATYPE element.

Paragraphs may not be nested. Embedded paralevel objects (like footnotes) are handled via the NOTE element described later.

Note that a PARA/HEAD element that has a parastyle will have PLF/CLF attributes *only* in cases of ***override***. Thus, in documents which were authored with heavy and consistent use of paratypes, the typical PARA/HEAD element will have no formatting attributes.

The content of a paragraph is stored in PARACONT elements. Any "leading adornment" is stored in the optional MARK subelement.

If the Rainbow generator encounters a contiguous sequence of one or more intra-paragraph line-breaks (often called a "hard-returns") in the source document, it should end the current PARACONT element and start a new PARACONT element with a "preced-hard-returns" attribute that is equal to the number of hard returns in the sequence.

If the Rainbow generator encounters a contiguous sequence of one or more tabs in the source document, it should end the current PARACONT element and start a new PARACONT element with a "preced-tabs" attribute that is equal to the number of tabs in the sequence.

It is legal for a PARACONT to have non-zero values for both of its "preced..." attributes.

```
<!ELEMENT (head|para)  - - (MARK?,PARACONT+)>
<!ATTLIST (head|para)
                       paratype CDATA    #IMPLIED
                       id       CDATA    #IMPLIED>
                       %plf-att; %clf-att;>

<!ELEMENT (paracont|mark)    - -
           (#PCDATA|CLF|SYSOBJ|ANCHOR|XREF|AUTOGEN|GRAPHIC|EQN|NOTE)+ >
<!ATTLIST paracont
                       preced-tabs         NUMBER #IMPLIED
                       preced-hard-returns NUMBER #IMPLIED >
<!ATTLIST mark         %clf-att;>

-------------------------------------------
```

Re: textual data: Any "non-Latin" character, or any "Latin" character whose integral representation exceeds 127, must be represented in Rainbow documents via references to entities, using standard ISO public entity sets. This ensures portability of Rainbow documents across Mac, PC, UNIX.

```
-------------------------------------------
```

If an equation or graphic occurs at the paralevel (e.g. a direct child of a section container), it must be bundled as an ILLUS element. If it is embedded in a paralevel object, it is simply represented as a GRAPHIC or EQN. In the former case, it is possible that there is an associated caption for the illustration. In Rainbow, that caption would simply be a PARA that lies before or after the ILLUS.

Consider a graphic embedded in an MS Word document. In the RTF, the graphic's binary data is represented via ASCII characters, with leading attributes specifying the binary format type (e.g. "Macintosh PICT"). The Rainbow extractor should perform the necessary conversions to create a file containing the graphic and generate a GRAPHIC element with appropriate filename and format (e.g. "PICT") attributes.

Equations are simply another form of illustration object, but it is a bit more flexible in that the data can be stored either in a file or as CDATA in the EQN element itself. Consider an equation represented in MS Word's proprietary equation language. The Rainbow generator could place the equation (still in MS's proprietary language) in a file and create a EQN element with format="msword-equ" and an appropriate setting for the filename attribute. Alternatively, the data could be placed in the EQN element and the filename attribute removed. A third alternative is that the Rainbow generator convert the equation to the ISO-standard equation DTD and store that in the EQN element and set format to "iso-dtd".

Note that the values for the format attribute have not been determined (the above examples used fictitious ones). Third parties implementing Rainbow generators should consult with EBT to "register" standard format names for both graphics and equations.

```
<!ELEMENT illus     - - (GRAPHIC|EQN)>
<!ATTLIST illus        %plf-att;
                       id       CDATA #IMPLIED>

<!ELEMENT graphic   - O EMPTY>
<!ATTLIST graphic      format CDATA #REQUIRED
                       filename CDATA #REQUIRED>

<!ELEMENT eqn       - - CDATA>
<!ATTLIST eqn          format CDATA #REQUIRED
                       filename CDATA #IMPLIED>

-------------------------------------------
```

CLF elements may not be nested. Thus, the sequence "Use ***Dyna*Text** for" would be represented this way:

```
Use <CLF weight=bold slant=ital>Dyna</><CLF weight=bold>Text</> for
```

```
<!ELEMENT clf       - - (#PCDATA|SYSOBJ|AUTOGEN)+>
<!ATTLIST clf          %clf-att;
                       ID CDATA #IMPLIED >

-------------------------------------------
```

NOTE elements store the content of objects like footnotes, annotations and sidebars; anything that would be represented in an on-line document as a "revelatory". In some WPs, the contents of notes are not stored directly at the point where the note is "referenced" in the text; but in Rainbow, the NOTE element must occur at the reference point. The Rainbow generator is responsible for performing any necessary "data movement".

```
<!ELEMENT note - - ((PARA|ILLUS|TABLE|SYSOBJ)*)>

-------------------------------------------
```

Some word processors can auto-create "back-of-the-book" indexes. The author of the document is responsible for marking words and phrases that should appear in the index. Rainbow generators should mark these index entries using the IDXTERM element.

If part of the document prose is marked as an index term, the IDXTERM element is used with no attribute:

```
A <IDXTERM>manatee</> is a...
```

If the index term (e.g "mammals, interesting") is not actually represented in the prose, the IDXTERM element is used with its "term" attributes:

```
A manatee <IDXTERM term1=mammals term2=interesting> is a...
```

Note that three term attributes are available, allowing up to three levels of hierarchy in the index. Note also that if any term attribute is set on a particular IDXTERM element, the PCDATA content of that element is ignored.

Some WPs allow the author to specify that an index entry should reference a different entry (*Amphibian: see Mammal*); this is represented in the Rainbow via the "see" attribute. Still others support similar "see also" semantics, represented in the Rainbow via the "see-also" attribute.

```
<!ELEMENT idxterm    - - (#PCDATA|CLF)>
<!ATTLIST idxterm       term1 CDATA #REQUIRED
                        term2 CDATA #IMPLIED
                        term3 CDATA #IMPLIED
                        see CDATA #IMPLIED
                        see-also CDATA #IMPLIED>

-------------------------------------------
```

Some word processors support the specification of cross-references. I.e. the author can attach "bookmarks" (IDs) to certain points in the document, and reference those bookmarks arbitrarily throughout the document.

A Rainbow generator encountering a bookmark should do one of these actions:

- If the bookmark is intended to give an ID to an entire PARA, CLF, HEAD, ILLUS, or TABLE, the bookmark's ID should simply become the ID of the PARA, CLF, HEAD, ILLUS, or TABLE in the Rainbow document.
- If the bookmark is intended to give an ID to a chunk of text that is embedded in a paralevel object (e.g. a word or phrase inside a paragraph), that chunk of text should be contained within an ANCHOR element in the Rainbow document. (Note that if the chunk of text that is being marked crosses one or more paralevel boundaries, this use of ANCHOR is not possible. See the next bullet.)
- If the bookmark is intended to mark a specific point in the document, an ANCHOR element with no content should be inserted in the Rainbow document at that point.

There are three types of word-processor-based "cross-references" which refer to bookmarks:

- The author inserts a reference to a bookmark in order to request that the word-processor auto-generate a page number. For example:
  `For more information, see the note on page [PAGEGEN-XREF TO BOOKMARK XX].`
  The Rainbow generator should generate a no-content XREF element with type set to "xref-pagenum".
- The author inserts a reference to a bookmark ***that contains textual content*** in order to request that the word-processor copy that textual content into the document at the current point. For example:
  `For more information, see the section called [TEXTCOPY-XREF TO BOOKMARK XX].`
  The Rainbow generator should generate a no-content XREF element with type set to "xref-textcopy".
- The author inserts a link, because she is using a word processor that is modern enough to support hypertext navigation. The Rainbow generator should generate an XREF element with type set to "xref-link".

```
<!ELEMENT anchor     - - (#PCDATA|CLF)>
<!ATTLIST anchor          ID CDATA #REQUIRED>

<!ELEMENT xref       - - (#PCDATA|CLF)>
<!ATTLIST xref            REFID CDATA #REQUIRED
                          TYPE (xref-textcopy|xref-pagenum|xref-link) #REQUIRED>

-------------------------------------------
```

Most WPs support autonumbering; in the WP file, a special object is placed at the point where the auto-generation is to occur. Rainbow generators are responsible for performing any work necessary to compute the ***resultant text*** that would actually be generated by the WP. That resultant text should be placed inside an AUTOGEN element to record the fact that the text is not source material. In the future, the AUTOGEN element's attribute list may be enhanced to allow storage of the algorithm used in the autogeneration.

```
<!ELEMENT autogen       - - (#PCDATA)>

-------------------------------------------
```

If it can do so, the Rainbow generator must produce empty NEWPAGE elements announcing the beginning of each page that would be generated if the user were to print the original source document. The page label is the the page identification that the user would see on the printed page; it is arbitrary character data in order to handle the wide variation in "page labelling" methods (e.g. Roman, number/letter combinations, etc.).

The seqnum attribute must specify the actual page number (which is not necessarily the same as the label that the user would see) to facilitate sending messages back to the word processor to request display of this page. The first physical page must have a seqnum of 1, the second a seqnum of 2, etc.

Note that a NEWPAGE element must be generated for each new page, even if the page starts with an object whose paratype specifies a page-break.

```
<!ELEMENT newpage       - O EMPTY>
<!ATTLIST newpage       pagelabel  CDATA #REQUIRED
                        seqnum     NUMBER #REQUIRED>
-------------------------------------------
```

The SYSOBJ is used to represent WP-specific data not covered by any standard Rainbow elements.

Third-parties building Rainbow generators should consult with EBT regarding each intended use of SYSOBJ.

We invite commentary from third-parties re: SYSOBJ syntax and semantics; we are open to modifications on anything regarding SYSOBJs.

```
<!ELEMENT sysobj   - O (datafld|txtfld)+>
<!ATTLIST sysobj      type CDATA #REQUIRED
                      id       CDATA #IMPLIED>
<!ELEMENT datafld  - - CDATA>
<!ELEMENT textfld  - - (#PCDATA|CLF)+>
<!ATTLIST (datafld|textfld)   type CDATA #REQUIRED>

-------------------------------------------
```

The CALS DTD is the model we followed to create this simple DTD representation:

```
<!ELEMENT table    - O (tgroup+)>
<!ATTLIST table       %plf-att;
                      id CDATA #IMPLIED>
<!ELEMENT tgroup   - - (colspec*,spanspec*,thead?,tbody)>
<!ATTLIST tgroup      cols NUMBER #REQUIRED>
<!ELEMENT colspec  - - EMPTY>
<!ATTLIST colspec     colnum NUMBER #IMPLIED
                      colname NMTOKEN #IMPLIED
                      align (left|right|center) #IMPLIED
                      colwidth CDATA #IMPLIED>
<!ELEMENT spanspec - - EMPTY>
<!ATTLIST spanspec    namest NMTOKEN #REQUIRED
                      nameend NMTOKEN #REQUIRED
                      spanname NMTOKEN #REQUIRED
                      align (left|right|center) #IMPLIED>
<!ELEMENT thead    - - (row+)>
<!ELEMENT tbody    - - (row+)>
<!ELEMENT row      - - (entry+)>
<!ELEMENT entry    - - (PARA|ILLUS|SYSOBJ)*>
<!ATTLIST entry       colname NMTOKEN #IMPLIED
                      spanname NMTOKEN #IMPLIED>

-------------------------------------------
```

5. Acknowledgements

The primary author of the Rainbow DTD and this document was David Sklar, who is grateful for the contributions made by Ludo van Vooren, Bobby Koritala, Brian Adler, Steve deRose, Lou Reynolds, and the many reviewers and supporters.

6. How to keep in touch with Rainbow

EBT is making the Rainbow DTD and various Rainbow-Maker software freely available to the public via an anonymous FTP server directory: ftp.ebt.com:/pub/outgoing/rainbow

If you wish to be notified by email when information in the FTP directory is modified and when significant Rainbow-related events occur, subscribe to the Rainbow information service by emailing to: rainbow@ebt.com