Internet Engineering Task Force
INTERNET-DRAFT
Expiration Date: 29 June 2000

R. Shirey
GTE / BBN Technologies
29 December 1999

The UGLI Method: Using Gargantuanware to Layout Internet Documents <draft-shirey-ugli-01.txt>

Status of this Memo

Copyright Notice

Abstract

The ASCII-text file format of Request for Comments (RFC) documents was devised decades ago when the authoring tools were line editors like vi and text processors like nroff. The format still has great utility because it meets the constraints of many kinds of printing and display equipment, but it is difficult to produce with some of today's desktop publishing tools. This paper describes the RFC text file format and discusses its fine points; this should be useful to authors of RFCs and Internet Drafts regardless of their tools. This paper also describes the UGLI method, a hybrid method for producing RFCs by using modern software for composition and editing and using

older software, applied in "cookbook" fashion, to finish the job. The method is described in terms of Microsoft Word on an Apple Macintosh but is easily adaptable for other word processors on other platforms.

Table of Contents

Table of Figures

Table of Fragments

Table of Tables

1. Introduction

The Request for Comments (RFC) document series is the official publication channel for Internet Standards and other Internet community information [R2026]. RFC 2223 [R2223] tells how to format and submit an RFC. Every RFC must be available in ASCII text. Some RFCs are also available in a secondary format, PostScript, but the ASCII version is the definitive reference.

The RFC series began in 1969, and its easy-to-read, time-tested file format was codified soon after. At that time, the Internet community produced documents mainly by using line editors (e.g., vi) to create input files for nroff [Ossa] and similar programs that format text for typesetter- and typewriter-like terminals. Today, people use sophisticated word processing software, such as Microsoft Word, Corel WordPerfect, Lotus Word Pro, and Adobe PageMaker. Surprisingly, although RFC format was fairly easy to produce with the older tools,

it often is difficult or impossible to produce with newer desktop publishing systems.

This paper describes a method for producing RFC format using a hybrid of old and new tools. It is called Using Gargantuanware to Layout Internet Documents (UGLI), where "gargantuanware" refers to the large, elaborate desktop publishing tools such as Word that are overkill for producing the RFC format. As suggested by the acronym "UGLI", the method is not pretty, but it does the job well. UGLI is described here for Microsoft Word on a Macintosh, but the method is easily adaptable for other word processors and platforms. It uses a modern word processing tool for the composition and editing work and then uses older tools, nroff and vi, for the final steps. This paper assumes that you know how to use Word or some equivalent tool. However, you do not need to know nroff or vi; complete step-by-step instructions are provided here for using them.

1.1 The Problem

As RFC 2223 says, "Online RFC files are copied by interested people and printed or displayed at their site on their equipment. This means that the format of the online files must meet the constraints of a wide variety of printing and display equipment." Thus, the primary format for RFCs remains ASCII text. Also, for comprehensibility, esthetics, readability, and tradition, "The RFC Editor attempts to ensure a consistent RFC style"; and that style involves typography that originated with nroff.

The RFC format is also used for other Internet Standards Process [R2026] documents (ISPDs). Prior to publication of an RFC, the material is made available to the Internet community as an Internet-Draft (I-D), which also is formatted as an ASCII text file. Most I-Ds appear in RFC format because their authors usually intend them to become RFCs. However, greater freedom is permitted in formatting I-Ds than RFCs. Consequently, many I-D authors avoid

full compliance with RFC 2223, apparently because the RFC requirements are difficult to satisfy.

Section 2 of this paper describes in detail the format for ASCII text RFCs. Both the requirement for ASCII text and the specified typography cause trouble for Word. Despite the simplicity of the RFC format, some features are impossible to achieve in an ASCII

text file by using only Word. At its heart, Word assumes that a document file will be displayed on the computer's screen for WYSIWYG editing and then will be printed on paper. Word maintains a specially formatted file for displaying the document, but that formatting is lost if the document is saved as a text file.

For example, RFC sections and subsections must be decimal numbered in outline style as "1.", "1.1", "1.1.1", and so on. Headings and paragraphs are stairstep indented by three spaces for each level, as illustrated by this I-D. Word has "style" features for automatically numbering and indenting for display or printing, but that formatting is lost if the author does a "Save As" to create a "Text Only" or "Text Only with Line Breaks" file.

For some lost formatting, there are easy workarounds. For example, the author can number sections manually. Revisions may force renumbering, but that is not too laborious except in very large, complex documents. Also, renumbering really needs to be done only once, in the final draft.

For some other lost formatting, such as stairstep indenting, available workarounds are not practical. For example, in the text file for this I-D, each line of this paragraph must begin with six blank characters, but Word does not put in leading blanks when saving indented material to a text file. The author can put in leading blanks before saving the file as text, but then the author loses many advantages of using Word for editing, such as being able to globally substitute one word or phrase for another. Substituting in a line typically changes where the ends of many of the following lines occur in the same paragraph, and where the ends of pages occur. This forces the author to manually remove and reinsert the sets of leading blanks and the page headers and footers, which is prohibitively time-consuming and error-prone.

For still other lost formatting, there is no workaround in Word. For example, each RFC page must be limited to 58 lines plus a form feed (FF, ASCII decimal 12) on a line by itself. Word has features to automatically paginate for display or printing, or the author can instead manually insert "Page Break" symbols. However, all pagination markers, both automatic (soft) and manual (hard), are lost when the author uses "Save As" to create a text file.

Mike Gahrns and Tony Hain of Microsoft Corporation specified a method for using Word 97 to produce a file that satisfies RFC 2223, but their method does not apply widely. First, it is

specific to Word 97, although Gahrns and Hain say it should be possible to "create a similar template for other versions" of Word. Second, the method depends on output from a generic text printer driver. Such a driver is not available with Macintosh Word and may not be in other environments. Third, the method requires an additional Win16/32 executable program to correct the output from the printer driver so that each line and each page terminates with the control characters required by RFC 2223. In short, their method is not easily portable across the wide range of desktop publishing environments found in the Internet community.

1.2 An UGLI Solution

RFC 2223 says, "RFCs in ASCII Format may be submitted to the RFC Editor ... in either the finished publication format or in nroff", and, "The RFC Editor may choose to reformat the RFC submitted."

When you take pride in your work, you prefer that no one reformats that work or otherwise changes the final version. Therefore, you submit your RFC in the finished publication format. If you do not, it is probably because you don't know how to achieve that form. UGLI solves that problem for you.

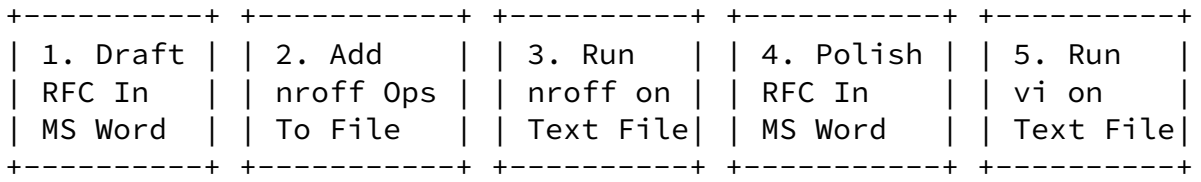
UGLI is applicable, or at least adaptable, to a wide range of desktop publishing environments. In UGLI, you first use Word or a comparable word processor for things that modern word processors do well, such as outlining, typing with automatic correction, cutting and pasting, searching and replacing, and checking spelling and grammar. Next, you enhance the Word file and create a text file for input to nroff. This paper assumes that you know how to use your word processor, but does not assume that you know nroff. Step-by-step, "cookbook" instructions are provided here for creating RFC format with nroff. After running nroff, you again use Word for some polishing, and then briefly use vi or a comparable text editor to insert control characters needed in the final file format of an RFC. You only need to type three vi commands, for which complete instructions are provided here.

Authors that use a Macintosh or Windows operating system normally do not have nroff or vi on their platforms. However, they usually can access a UNIX host, and nearly every UNIX host has nroff and vi available. The UNIX host that this author uses for UGLI is his departmental POP3 host, the one from which Eudora retrieves his mail. The user identifier and password that Eudora uses to retrieve mail provides the account for running nroff and vi.

1.3 Five UGLI Phases

UGLI has five phases, as shown in Figure 1. This section briefly describe the phases; later sections discuss them in detail.

Figure 1. UGLI Phases



In Phase 1, you compose your RFC in Word. Section 2 describes the RFC format, and Section 3 provides guidance for configuring a Word file to resemble a finished RFC, so that you can see approximately what your final product will look even while you are still working in Word. However, if you save that Word file as a text file, the RFC formatting done by special Word features will be lost.

In Phase 2, therefore, you enhance the Word file by inserting nroff commands (called "requests") that will create the RFC format after you save the file as a text file. Section 4 provides "cookbook" directions and examples for nroff. The Word formatting established in Phase 1 provides a visual guide for inserting the correct requests. Also, the requests called for by Section 4 are simplified so that debugging work should be minor or non-existent.

You should format nroff requests as "hidden text" in Word, so that you can hide them when you want to print or view a rough draft of the RFC directly from Word without going through nroff. You can create this "dual use/dual view" Word/nroff file by performing Phases 1 and 2 serially; or you can do them in parallel, putting in nroff requests at that same time that you compose and edit the RFC text. Then, you save the Word file, including the embedded nroff requests, as a text file, which becomes nroff's input file.

In Phase 3, you move that file to a UNIX host, and you run nroff on that file. You only need to type one simple nroff command.

In Phase 4, you move nroff's output back to Word and review the result. If changes are needed, you return to Phase 1 or 2. If not, you do cleanup editing in Word, including putting page numbers into the RFC's table of contents and getting ready to process the file with vi. You then save the Word file as a text file, which becomes vi's input file for the next phase.

In Phase 5, you move the file to a host that has vi or a similar line editor program. You use vi to put a form feed character on the last line of each page. You only need to type three simple vi commands. You then move vi's output file, which now completely satisfies the requirements of RFC 2223, back to where you have Word, so that you can easily review and print the file.

2. Format Requirements for RFCs

This section is *not* the official specification of RFC format; for that, see section 3 RFC 2223. But this is a more complete specification. First, this section points out some less-than-obvious implications and nuances of requirements that RFC 2223 states explicitly. Second, this section explicitly states requirements that RFC 2223 only implies. Third, this section addresses some topics that RFC 2223 ignores completely. The key words "MAY", "MUST", "MUST NOT", and "SHOULD" are intended to be interpreted the same way as in an Internet Standard [R2119], but they represent only this author's interpretations of RFC 2223.

- RFC 2223 states explicit requirements for characters, lines, and pages. Here, Section 2.1 explains those rules and suggests revisions and extensions to improve RFC appearance and efficiency.

- RFC 2223 states explicit requirements for page headers and footers. Section 2.2 explains those rules.

- RFC 2223 says, "Please do look at some recent RFCs and prepare yours in the same style." That implies requirements for section numbering and indenting. Section 2.3 states those rules.

- RFC 2223 neither specifies nor implies formatting for many other aspects of an RFC, such as how to handle "bullets" and exhibits (figures, tables, and so on). Section 2.4 makes suggestions for

those things, and this I-D provides examples for them and for all the other format features mentioned in this section.

2.1 Explicit Requirements for Characters, Lines, and Pages

Section 3a of RFC 2223 states text format requirements, which this section reorders and numbers as eight "rules". (RFC 2223 also bans footnotes and recommends that cross-references point to section numbers instead of page numbers, but those constraints do not affect UGLI and are not discussed further in this I-D.)

- Rule 1: "The character codes are ASCII [ANSI]."
- Rule 2 (Width): "Each line must be limited to 72 characters followed by carriage return [CR, ASCII decimal 13] and line feed [LF, ASCII decimal 10]." This could be restated as follows: each line MUST contain at least 2 and at most 74 characters, and the last two on each line MUST be CR and LF.
- Rule 3 (Height) : "Each page must be limited to 58 lines followed by a form feed [FF, ASCII decimal 12] on a[n additional] line by itself."
- Rule 4: "[The foregoing] 'height' and 'width' [rules] include any headers, footers, page numbers, or left side indenting."

- Rule 5: "No overstriking (or underlining) is allowed."
- Rule 6: "Use single spaced text within a paragraph, and one blank line between paragraphs."
- Rule 7: "Do not fill the text with extra spaces to provide a straight right margin."
- Rule 8: "Do not do hyphenation of words at the right margin."

The following interpretation and analysis of these rules reveals problems that lead to suggestions for revising some of them.

2.1.1 Placement of Form Feed Characters

The first four rules are not entirely simple to satisfy. First,

notice that Rule 3 applies to *every* page, including the last page. Next, looking at Rule 3 alone, you might think that it has two equally correct interpretations:

- Interpretation 1 for Rule 3:

```
[Line m   of page n  ]  Last  non-blank line of page. CR LF
[Line m+1 of page n  ]  FF
[Line 1   of page n+1]  First non-blank line of page. CR LF
```

- Interpretation 2 for Rule 3:

```
[Line m   of page n  ]  Last  non-blank line of page. CR LF
[Line m+1 of page n  ]  FF CR LF
[Line 1   of page n+1]  First non-blank line of page. CR LF
```

The combination of Rules 2 and 3 implies that interpretation 1 is incorrect; you don't get another "line" unless you have a line feed. Interpretation 2 is correct; a line containing the FF character (the last line of a page) must be terminated by CR and LF characters, just like all other lines on a page.

Interpretation 2 has somewhat unexpected consequences when you view an RFC. When a file satisfies Rule 3 and you display or print that file with Word, you might suppose that you get 58-line pages separated by Word "page breaks". Instead, assuming that each page has the maximum number of lines permitted by Rule 3, you get the following:

- Page 1 prints only 58 lines, from the first, non-blank line of the header to the last, non-blank line of the footer. That is because the first character on the 59th line of the file (i.e., the 59th line of page 1, as viewed by Rule 3) is an FF. So right after the 58th line, the printer begins a new page, page 2.

- Next, more surprisingly, the 59th line of the file also causes a blank line to print at the top of page 2. That is because immediately after the FF there is a CR that prints nothing visible and an LF that shifts printing to the next line, the second line of the page, where the first, non-blank header line of page 2 is printed.

- o Similarly, a blank line is printed at the top of every page after page 1, and a final blank page (consisting of one blank line and nothing else) is printed after the final numbered page of the RFC.

To summarize, when you display an RFC that conforms with Rules 2 and 3, page 1 is seen to have 58 (or fewer) lines (counting from the first, non-blank line of the header to the last, non-blank line of the footer). All other numbered pages have one more line than page 1; they start with a blank line and continue with 58 (or fewer) more lines like on page 1. Thus, when you print, you get the ugly result that page 1 starts one line closer to the top edge of the paper, and ends one line farther away from the bottom edge, than do the other pages.

The difference between page 1 and the other pages is hardly noticeable if you read the text on a screen or print only one copy with one RFC page per page of paper. But the difference becomes noticeable if you print camera copy for a multi-copy distribution or print two RFC pages per paper page in order to save space, weight, or trees. (Have you noticed the difference before? It should occur regardless of whether Word or another tool is used for printing.)

UGLI's output should not be "ugly". Therefore, Phase 4 includes a step to "correct" your RFC by adding a blank line at the beginning, so that page 1 will display just like all the other pages do. This I-D has been repaired in that way. (Of course, it would be better if the RFC Editor would change the rules to effect interpretation 1 for Rule 3.)

2.1.2 Number of Lines per Page

The combination of Rules 3 and 4 leaves a loophole that can result in truly ugly RFCs because of "bouncing footers". Rule 4 includes the footer in the line count, but Rule 3 permits the number of lines per page to vary (as seen in several existing RFCs). In that case, the footer (which always contains visible, non-blank characters, as specified by Rule 12 in Section 2.2) "jumps up and down", which is unpleasant to see. Therefore, if and when RFC 2223 is superseded, the RFC Editor should stiffen Rule 3 as follows:

- Rule 3A: Each page MUST have exactly the same number of lines, which MUST be limited to (n <=) 58 lines followed by a form feed (FF, ASCII decimal 12) by itself on another line (line n+1).

UGLI, as illustrated by this I-D, enables the RFC author to follow an even stiffer version of the rule:

- Rule 3B: Each page MUST have exactly 59 lines, where the 59th line contains only the characters FF, CR, and LF, in that order.

If the author chooses to "repair" page 1 (as explained in step "B" in Section 5.4), then Rule 3 effectively becomes restated as follows:

- Rule 3C: Page 1 MUST begin with a blank line and have exactly 60 lines. All other pages MUST have exactly 59 lines. The 60th line of page 1 and the 59th line of other pages contain only FF, CR, and LF, in that order.

2.1.3 ASCII Characters

Rule 1 means that all the character codes in an RFC text file must be from the ASCII set defined in [ANSI]. The set of 128 ASCII codes includes the 33 control characters shown in Table 1 and the 95 graphic characters shown in Table 2.

Table 1. ASCII Control Characters

Dec	Chr	Meaning	Dec	Chr	Meaning
000	NUL	null	016	DLE	dev link esc
001	SOH	start header	017	DC1	dev ctrl 1
002	STX	start text	018	DC2	dev ctrl 2
003	ETX	end text	019	DC3	dev ctrl 3
004	EOT	end of transmit	020	DC4	dev ctrl 4
005	ENQ	enquiry	021	NAK	negative ack
006	ACK	acknowledge	022	SYN	sync idle
007	BEL	bell (beep)	023	ETB	end trans block
008	BS	back space	024	CAN	cancel
009	HT	horiz tab	025	EM	end medium
010	LF	line feed	026	SUB	substitute
011	VT	vert tab	027	ESC	escape
012	FF	form feed	028	FS	cursor right
013	CR	carriage ret	029	GS	cursor left

014 SO	shift out	030 RS	cursor up
015 SI	shift in	031 US	cursor down
		127 DEL	delete

The control characters were designed for operating remote teletype equipment. (RFC 854 [R854] discusses the function of some these characters in the context of the TELNET network virtual terminal.) Rules 2 and 3 require the FF, CR, and LF characters; and Rule 5 prohibits some types of typography that would use the BS character. Other than that, RFC 2223 does not mention control characters or possible "rogue" uses that would clearly be inappropriate in an RFC. For example, what if someone wants to have fun by including the BEL character? That would not "meet the constraints of a wide variety of printing and display equipment".

So, if and when RFC 2223 is superseded, the RFC Editor should consider this stiffened version of Rule 5:

- Rule 5A: Except for CR and LF at the end of each line, and FF on the last line of each page, each character MUST be one of the 95 ASCII graphic characters (which are shown in Table 2 with the decimal values of their ASCII codes).

Table 2. ASCII Graphic Characters

Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char
032	[space]	048	0	064	@	080	P	096	`	112	p
033	!	049	1	065	A	081	Q	097	a	113	q
034	"	050	2	066	B	082	R	098	b	114	r
035	#	051	3	067	C	083	S	099	c	115	s
036	$	052	4	068	D	084	T	100	d	116	t
037	%	053	5	069	E	085	U	101	e	117	u
038	&	054	6	070	F	086	V	102	f	118	v
039	'	055	7	071	G	087	W	103	g	119	w
040	(	056	8	072	H	088	X	104	h	120	x
041	)	057	9	073	I	089	Y	105	i	121	y
042	*	058	:	074	J	090	Z	106	j	122	z
043	+	059	;	075	K	091	[	107	k	123	{

```
 044 ,       060 <       076 L       092 \       108 l       124 |
 045 -       061 =       077 M       093 ]       109 m       125 }
 046 .       062 >       078 N       094 ^       110 n       126 ~
 047 /       063 ?       079 O       095 _       111 o
-------------------------------------------------------------------
```

2.1.4 Blank Lines

Rule 6 is incomplete or imprecise. A blank line is needed not only between two paragraphs but also between a heading and a paragraph, between an exhibit and a paragraph, and in other places. Also, the term "blank line", if restricted only by Rule 2, may be variously interpreted as zero to 72 blank characters (ASCII decimal 32) followed by CR and LF, even though no blank characters are actually needed to display or print a blank

line. Therefore, if and when RFC 2223 is superseded, the RFC Editor should consider the following revision of Rule 6:

- Rule 6A: A "blank line" is a line containing only two characters: CR and LF. Blank lines are placed as follows:

  A. The document title, which MAY consist of one or more lines, SHOULD be preceded by two blank lines to separate it from the header of page 1, and SHOULD be followed by two blank lines to separate it from the first heading, which is "Status of This Memo".

  B. Each heading, regardless of indent level, SHOULD be followed by one blank line.

  C. Each paragraph of text, regardless of indent level, SHOULD be followed by one blank line. Paragraph text MUST be single-spaced (i.e., there MUST NOT be a blank line between two non-blank lines of the same paragraph).

(Other uses of blank lines and other aspects of the format of exhibits are discussed in Section 2.4.)

For both display and printing, Word has features for automatically adding blank space for vertical separation. However, this "style" treatment is lost when a Word file is saved as a plain text file. Therefore, UGLI requires authors to

manually insert the required blank lines, and this is easily done by striking the return key.

2.1.5 Right Page Margin

There are good reasons why Rule 7 requires the right page margin to be "ragged", and why Rule 8 prohibits hyphenating words at the right margin. First, studies have shown that text is harder to read when both margins are adjusted than when just the left margin is adjusted, regardless of which font is used or how smoothly blank filler is inserted. Second, when technical text in a fixed-width font is hyphenated at the right margin, the printed result is not only less readable but also ugly (which is not the same as UGLI).

2.2 Explicit Requirements for Page Headers and Footers

Section 4 of RFC 2223 states the following explicit requirements for page headers and footers, and for page numbers, which are included in the footers:

- Rule 9: All pages MUST be numbered consecutively, beginning with "Page 1". (A page's number appears in the page's footer, as specified in Rule 12 below).

- Rule 10: Page 1 MUST have a special, multi-line, variable-length header, which is specified in Section 4a of RFC 2223 and illustrated in this I-D. (In UGLI, the RFC author composes page 1's header as described here in Section 4.2.)

- Rule 11: All pages except page 1 MUST have the same three-line running header, as specified in Section 4b of RFC 2223 and illustrated in this I-D:

  - RFC 2223 explicitly specifies only the first line of the running header. The first line MUST have "the RFC number on the left (RFC NNNN), the (possibly shortened form) title centered, and the date (Month Year) on the right."

  - RFC 2223 implies that the running header MUST be three lines long, where the second and third lines are blank.

- o Rule 12: All pages, including page 1, MUST have three-line, footers that, except for the page number, have identical content, as specified in Section 4c of RFC 2223 and illustrated in this I-D:

  - RFC 2223 explicitly specifies only the last line of the footer. That line MUST have "the author's last name on the left, category centered, and the page number on the right" in the form "[Page N]".

  - RFC 2223 implies that the footer MUST be multi-line, with some number of blank lines preceding the last, non-blank line.

    -- In many RFCs, the footer is four lines long; that is, at least three blank lines always precede the non-blank line that contains the page number.

    -- In this I-D, the footer is only three lines long, so that at least two blank lines always precede the single non-blank line that contains the page number. Here are some reasons for including only two blank lines in the footer:

    ---Paragraphs are separated by one blank line. Thus, even if a page is completely full, two blank lines provide sufficient (i.e., visually unambiguous) separation between the last non-blank line of text on a page and the non-blank (i.e., third) line of the footer on that page. (As discussed for enhancement "A" in Section 4.5, the last line of text on a page should be the last line of a paragraph or exhibit. It would ugly if it were a heading.)

    ---The non-blank (i.e., first) line of the running header is normally separated from a page's text by two blank

lines (but might sometimes be separated by three blank lines, as discussed in the last paragraph of Section 4.5.) So, using two blank lines in the footer lends a pleasingly symmetric appearance to full pages.

---Adding a third blank line to the footer would waste almost 2% of the available text space on many pages.

UGLI automatically determines where each page break should be; automatically inserts the 3-line footer for the end of the previous page, including the proper page number; and automatically inserts the 3-line header for the next page.

However, the page numbers must be manually copied into the Table of Contents as described in Section 5.4., and the LF characters on the last line must be added later as described in Section 5.5.

2.3 Implicit Requirements for Numbering and Indenting

RFC 2223 implies the following requirements for numbering sections and for indenting section headings and paragraphs:

- Rule 13: Except for some preamble sections and the abstract, sections MUST be decimal numbered in outline style: "1.", "1.1", "1.1.1", etc., as illustrated by this I-D.

For both display and printing purposes, Word include features that automatically number sections. However, this "style" treatment is lost when a Word file is saved as a plain text file. Therefore, UGLI requires authors to number sections manually. Some authors will find this laborious, but it is not onerous except in very large, complex documents. Also, it needs to be done only once, when putting on the finishing touches.

- Rule 14: Heading indents: Headings MUST be step-indented as follows, as illustrated by this I-D:

  1-st level heading (i.e., numbered "N.") is flush left.
  2-nd level heading (i.e., "N.N") is indented 3 spaces.
  i-th level heading is indented 3*(i-1) spaces.

- Rule 15: Paragraph indents: Paragraphs MUST be step-indented as follows, as illustrated by this I-D:

  1-st level paragraph (under "N." heading) is indented 3 spaces.
  2-nd level paragraph (under "N.N") is indented 6 spaces.
  i-th level paragraph is indented 3*i spaces.

Word has features to automatically indent headings and paragraphs. However, this "style" treatment is lost when a Word file is saved as a plain text file. Therefore, UGLI requires that an author manually insert nroff requests to indent headings and sections,

before the Word file is saved as text. This is the most ugly aspect of UGLI, but it is mitigated by having Word's automatic indenting features available as a powerful "visual guide" that makes it easier for you to do these manual insertions correctly the first time.

2.4 Other Requirements

RFC 2223 neither explicitly specifies nor clearly implies format requirements for many kinds of typograpical devices that are commonly found in RFCs. This section discusses three of these: bullet paragraphs, content tables, and exhibits. This section also comments on requirements beyond formatting.

2.4.1 Bullet Paragraphs

RFC 2223 specifies no format for "bullet" paragraphs, and no style predominates in "recent RFCs". The bullet styles used in this I-D are acceptable, but bullets that go deeper than two levels (e.g., those shown four levels deep under Rule 12) can be annoying ... and ugly. In any case, an RFC should be internally consistent in its use of bullets.

Word has features to automatically add bullets to paragraphs. However, this "style" treatment is lost when a Word file is saved as a text file. UGLI requires that an author insert bullets manually. This is not onerous; as described in Section 4.2, it takes only a few keystrokes for each bullet paragraph.

2.4.2 Content Tables

RFC 2223 implies that an RFC MUST include a "Table of Contents" section but specifies no format for it, and no style predominates in "recent RFCs". Word has features to automatically generate the table, but some or all of the automatically generated material is lost when the file is saved as text. Also, page numbers are likely to be different after the nroff processing in Phase 3 and must be corrected in Phase 4. The author probably will do less work overall by constructing the Table of Contents manually. Some authors may find this laborious but, like section numbering, it needs to be done only once when putting on finishing touches.

The style used for the Table of Contents in this I-D is acceptable, but an author may prefer to show a greater or lesser number of heading levels. There is a trade-off between including more levels to convey more information and including

fewer levels to make it easier to grasp the overall structure of the RFC. Two levels are usually sufficient, but this I-D includes three as an illustration.

2.4.3 Exhibits

RFC 2223 neither states nor implies rules for formatting exhibits (i.e., for figures, diagrams, or other illustrations; and for tables and other data displays).

Rule 14 for indenting headings and Rule 15 for indenting paragraphs create problems for formatting exhibits. For example, each exhibit could be indented to align with the paragraph that refers to it, but that could be visually choppy. There does not seem to be a nice style that avoids all problems. For example, this I-D uniformly indents all exhibits by six spaces, like a second-level paragraph, but such an exhibit can be at most 66 characters wide (72-character line length minus 6-character indent). So, in an exhibit like Fragment 2, this I-D is unable to show all 72 characters from a line of its own nroff input file (catch 72?); some blanks must be squeezed out.

If and when RFC 2223 is superseded, the RFC Editor should consider including these format recommendations:

- Each set of exhibits of the same type SHOULD be numbered sequentially.

- Each exhibit SHOULD be followed by one blank line. (This extends Rule 6A.)

- Each exhibit SHOULD begin with a title line. The title line SHOULD include the exhibit's number and MAY be centered, as illustrated in this I-D.

- One blank line SHOULD separate an exhibit's title from the exhibit's body. (This also extends Rule 6A.)

- When it is necessary to make clear which lines belong to an exhibit and which belong to the surrounding text (e.g., see

Fragment 2), the exhibit SHOULD have a delimiter line at its top and at its bottom (i.e., before the title and after the body). In that case, all other exhibits in the document also SHOULD have delimiter lines. All delimiter lines SHOULD be identical.

- Additional content tables MAY be provided for exhibits (e.g., Table of Figures, Table of Tables) if the author believes this would be useful to readers. This I-D provides such tables as illustrations.

2.4.4 Terminology

Finally, consistency in ISPDs means more than just consistent format; consistent terminology is equally or more important. To

avoid confusion, all ISPDs SHOULD use the same term or definition whenever the same concept is mentioned. To improve international understanding, ISPDs SHOULD use terms in their plainest, dictionary sense. ISPDs SHOULD use terms established in standards documents and other well-founded publications and SHOULD avoid substituting private or newly made-up terms. For guidance, authors should see available glossaries [R1208, R1983, Shir].

3. The Word View

In UGLI Phase 1, you use Word to compose your RFC like you would any document. By configuring Word for page dimensions, indenting, headers and footers, and other features of RFC format, you can approximate the appearance of a finished RFC and, therefore, see what the final UGLI product will look like while you are still working just in Word.

Section 2 has described RFC format in detail, and this section suggests how to configure Word for that format. (Actually, this section describes how the author configures Word 98 on his Macintosh, but the description should apply directly to other versions of Word.)

- Page width: Set the font to 10-point Courier with normal spacing; this yields 12 characters per inch. Set the line length to 6 inches; this yields 72 characters per line. For example, in Page Setup, select "Letter (Small)" and set left and right margins at

1.25 inches each.

- Page height: Define the special header for page 1, the 3-line running header for the other pages, and the 3-line footer for all pages. Then set the top and bottom margins to yield 58 lines per page beginning with the first, non-blank line of the running header and ending with the last, non-blank footer line. For example, in Word's Page Setup, set the top margin at 1.17 inches and the bottom at 1.00 inch, and set "From Edge" to 0.75 inch for the header and 0.5 inch for the footer.

- Text spacing and alignment: Select "single" line spacing. Set Word to align (i.e., justify) text only on the left margin. Turn off automatic hyphenation.

- Indenting and centering: Define and apply "styles" that do automatic indenting for each level of heading, paragraph, and bullet paragraph that you use, with hanging indents for bullet paragraphs. Use 0.25 inch for each 3 characters of required indentation. Define a centered style for, or individually apply centering to, the main title at the top of page 1 and the titles of exhibits.

- Pagination enhancement: Turn off "widow and orphan" control. If you intend to enhance pagination as described in Section 4.5, apply "keep together" to exhibits and apply "keep with next" to heading styles.

4. The nroff View

This section describes how UGLI uses nroff to generate the RFC format. This is illustrated with fragments from the nroff input file used to produce this I-D. Commands in the nroff language are called "requests". Requests consist of a period (".", in column 1) and two-character alphabetic codes, sometimes followed by parameters.

4.1 Basic Formatting

The first five lines of the nroff input file for this I-D contain the requests shown in Fragment 1. The requests implement the rules for RFC page length, line length, and hyphenation. (In Figure 1, the two-character escape sequence, \", begins a comment that continues to the end of that line. In this fragment and those that follow, the comments are not necessary for the nroff formatting and can be omitted from the nroff input file.)

```
-----------------------------------------------------------------
              Fragment 1. Specifying Basic Formatting

 .pl 10.0i \" Set page length.
 .ll 7.2i  \" Set length of lines.
 .lt 7.2i  \" Set length of 3-part titles in headers and footers.
 .hy 0     \" Turn off hyphenation of words at right margin.
 .ad l     \" Adjust left margin but not right margin.
-----------------------------------------------------------------
```

4.1.1 Page Length

The nroff request ".pl 10.0i" sets the page length to 10.0 inches.

Why does it say exactly 10.0 inches? Because on this author's machine, that value (in combination with the ".wh" request described below in Section 4.3) produces 58 lines for each page of the nroff output file, beginning with the non-blank line of the header and ending with the non-blank line of the footer. To those 58 lines, UGLI later adds (see Section 5.5) a 59th line that contains only a form feed (as Rule 3 requires).

Depending on how nroff's default output font and printer device parameters are set on your machine, "your actual mileage may vary". If so, adjust the parameter values on the ".pl" and ".wh" requests until you get 58 lines per page.

4.1.2 Line Length

The nroff request ".ll 7.2i" (i.e., 7.2 inches) sets the line length for a full line of text, i.e., a line constructed with no indent (see ".in 0" see in Section 4.4 below).

Why does it say exactly 7.2 inches? Because on this author's machine, nroff's default character width is 1/10 inch. Thus, this request sets the maximum line length to 72 characters followed by CR and LF.

Depending on how nroff's default output font and printer device parameters are set on your machine, "your actual mileage may vary". If so, adjust the parameter value on this request until you get at most 72 characters per line. For example, your default character width might be 1/12 inch (because 1/10 and 1/12 are the most popular settings); if so, use ".ll 6.0i".

The nroff request ".lt 7.2i" sets the line length for the three-part titles that are used to create the non-blank lines in headers and footers (see ".tl" requests in Section 4.3). The ".lt" request operates the same way as the ".ll" request.

4.1.3 Hyphenation and Adjustment

The nroff request ".hy 0" turns off nroff's automatic hyphenation at the right margin (as Rule 8 requires).

In nroff, "adjusting" refers to positioning text on a line (with or without "filling" the line, as discussed in Section 4.2) and then optionally aligning the text on the left margin, the right margin, both margins, or the center line, including inserting extra blank spaces if needed.

The nroff request ".ad l" turns on the "adjusting" function. The parameter "l" (that's a lower case "L") directs that only the left margin should be adjusted (as Rule 7 requires).

4.2 Text Filling and First Page Header

Fragment 2 shows the next four lines of the nroff input file. These requests turn off nroff's text "filling" function and define the header for page 1. (To be able to indent Fragment 2, some blanks had to be removed from the full, 72-character lines of the header; see page 1 of this I-D for the full-length lines.)

Fragment 2. Specifying First Page Header

```
.nf                              \" Turn off line-filling function.
Internet Engineering Task Force                          R. Shirey
INTERNET-DRAFT                                GTE / BBN Technologies
```

The nroff request ".nf" turns off nroff's text "filling" function, so that the strings of blanks shown in Fragment 2 are preserved in

the output file.

In nroff, "filling" refers to collecting words from text lines of the input file and assembling them into an output file line (with exactly one space between each pair of words) until one of the following conditions occurs:

- Case 1: An end-of-line (in Word, that's a "manual line break" or a "paragraph mark") is encountered in the input.

- Case 2: Some word does not fit within the line length established by the ".ll" request described in Section 4.1.2.

In either case, if hyphenation is turned off as it is in this I-D, then the current output line is terminated, and the next input word is used to begin the next output line. (In case 2, this input word is the one that did not fit within the previous line.)

When nroff is run, the filling function is turned on by default. So, in the nroff input file for this I-D, we place a ".nf" request just before the layout for the three-line header for page 1, as shown above in Fragment 2. If we did not turn off the filling function, nroff would squeeze together the words in the first page header and output the lines shown in Figure 2.

Figure 2. Result of Failure to Turn Off Filling

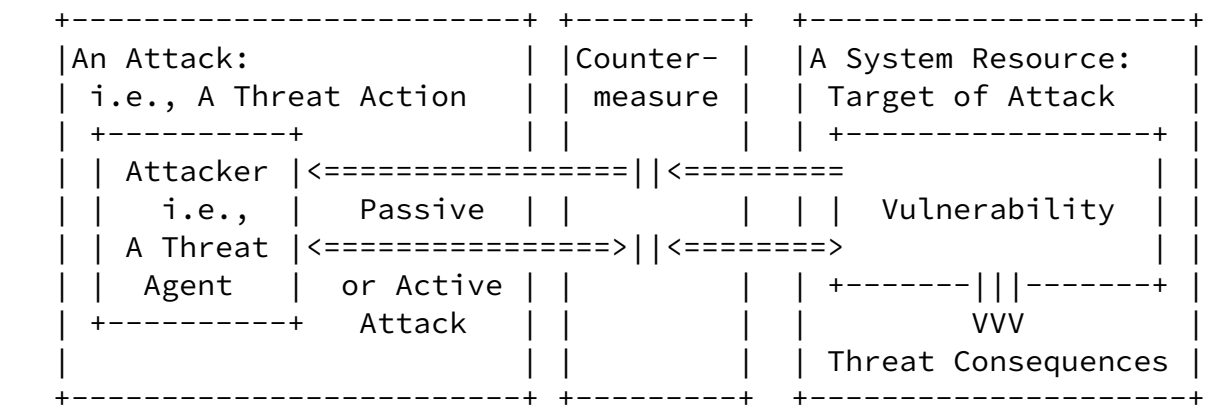
```
Internet Engineering Task Force R. Shirey
INTERNET-DRAFT GTE / BBN Technologies
Expiration Date: 22 June 2000 27 December 1999
```

The rest of the nroff input file that follows Fragment 2 keeps the filling function turned off at all times; that is, the request ".fi" is not used to turn filling on again. Thus, the only blank characters (ASCII decimal 32) in nroff's output file for this I-D

are (a) blanks that were explicitly placed in the input file and (b) leading blanks implied by indenting requests (see ".in" and ".ti" in Section 4.4) for section headings and paragraphs.

An author may choose to leave the filling function turned on or off. The advantage of keeping it turned off is that the formatting seen in nroff's input file is identical to what appears in the output file. Thus, it is easier to control the appearance of the output. Pieces of "ASCII art", such as the diagram shown in Figure 3, can be included without having to put ".nf" before each piece and ".fi" after each piece. The disadvantage of keeping the filling function off is that nroff does not remove extraneous spaces like these. You need to find and remove them yourself, but Word's "Find and Replace" function can help.

Figure 3. Sample of "ASCII Art"



4.3 Page Breaks

Fragment 3 shows the next eight lines of the nroff input file. These requests control page breaks, define the header for all pages except page 1, and define all footers. Comments in Fragment 3 (defined by a two-character escape sequence, \", and including the characters that follow to the end of a line) are optional.

Fragment 3. Specifying Footer, Page Break, and Running Header

```
.de NP                                          \" Define NP macro.
'sp 2                                           \" - 2 blank lines.
.tl 'Shirey'GTE / BBN Technologies'[Page %]'    \" - 1 non-blank.
.bp                                             \" - Begin page.
.tl 'Internet-Draft'UGLI'1 December 1999'       \" - 1 non-blank.
'sp 2                                           \" - 2 blank lines.
..                                              \" - End macro def.
.wh -.84i NP                                    \" Set trap for NP.
```

4.3.1 Macro Definition

The first seven lines in Fragment 3 define a macro. The macro definition has five parts: macro name, footer insertion, new page beginning, header insertion, and macro termination.

The request ".de" declares the beginning of the macro definition, and the parameter "NP" (for New Page) assigns a name to the macro.

The next two requests insert a three-line footer (as Rule 12 requires). The request "'sp 2" inserts two blank lines. Notice that the control character before the "sp" is an apostrophe ("'"), not a period (."). If you want to know why, please read

the manual [Ossa].

The request ".tl 'Shirey'GTE / BBN Technologies'[Page %]'" inserts a three-part title line in the current title length (which is set by the request ".lt" in Fragment 1):

- "Shirey" is left-adjusted.
- "GTE / BBN Technologies" is centered
- A bracketed page number is right-adjusted. (The default initial page number is "1", but the request ".pn N" can be used to set the numbering for the next page to N. This can be useful in assembling an RFC text file from parts that are processed as separate nroff inputs.

The request ".bp" directs nroff to begin a new page.

The next two requests insert a three-line header (as Rule 11

requires). The request ".tl 'Internet-Draft'UGLI'1 December 1999'" inserts a three-part title line in the current title length:

- "Internet-Draft" (or "RFC NNNN") is left-adjusted.
- "UGLI" is centered.
- "December 1999" is right-adjusted.

The request "'sp 2" inserts two blank lines, like in the footer.

The two character sequence ".." terminates the definition of the macro.

4.3.2 Setting a Trap to Invoke the Macro

The eighth line of Fragment 3 tells nroff when to execute the NP macro. The request ".wh -.84i NP" sets a trap at a position 0.84 inches above the bottom of each page.

Why exactly 0.84 inches? Because on this author's machine, that value (in combination with the page length request ".pl 10.0i" that is discussed in Section 4.1.1) produces exactly 58 lines on each output page beginning with the non-blank line of the header and ending with the non-blank line of the footer. However, "your mileage may vary." If so, adjust the parameters.

Each time that the trap is sprung by having a page become filled with text down to that position, nroff executes the NP macro. Also, the ".bp" request on the last line of the nroff input file, shown in Fragment 4, forces NP to execute so that that the last page receives a footer. This also places an extraneous header at the end of the nroff output file, but UGLI Phase 4 removes that header and does other cleanup tasks, too.

Fragment 4. Last Line of nroff Input File

```
.bp          \" Trigger NP macro to insert footer on last page.
```

4.4 Indenting, Centering, and Pagination

In the nroff input file for this I-D, Fragments 3 and 4 are separated by the body of the draft, i.e., the title, headings, paragraphs, and exhibits. This section explains nroff requests that are used to center the main title and the titles of exhibits, indent headings and paragraphs, and begin a new page when desired. Figure 4 illustrates these requests. Comments in Figure 4 are optional and may be omitted.

Figure 4. Examples of Centering and Indenting

```
 .in 0 \" Indent all output lines 0 spaces until further notice.
 [Two blank lines are placed here to separate the title from the
 3-line header on page 1.]
 .ce 2 \" Take next 2 input lines and center them in output file.
                    This Title Line Is Centered
                         and So Is This One
 [Two blank lines are placed here to separate the title from the
 first heading on page 1.]
 1. A Heading at Level One]
 [One blank line here.]
 .in 3 \" Indent all output lines 3 spaces until further notice.
 [One blank line here.]
    A paragraph at level one.
 [One blank line here.]
    1.1 A Heading at Level Two.
 .in 6 \" Indent all output lines 6 spaces until further notice.
 [One blank line here.]
       A paragraph at level two.
 [One blank line here.]
 .in 9 \" Indent all output lines 9 spaces until further notice.
 .ti 6 \" Indent next output line 6 spaces, then do as before.
       o  An illustration of how to create a hanging indent by
          using the ".ti" request.
 [One blank line here.]
 .bp   \" Begin new major section at the top of a new page.
 2. Another Heading at Level One]
 [One blank line here.]
```

An nroff request ".in N", where N is an integer, sets the indent depth to be N spaces from the left margin for lines in the nroff output file. The setting is "permanent"; it applies to all lines generated after the request until another ".in" or ".ti" request

is encountered. As shown in Figure 4, the indent for text headings and paragraphs should be set as specified by Rules 12 and 13 in Section 2.3.

The ".in 0" request that begins Figure 4 is unnecessary, because the default initial indent setting is "0". The request is included here as a reminder to authors that they must properly indent all lines in the rest of the document.

The nroff request ".ti N" sets the indent depth for the next output line. The setting is "temporary"; it applies only to that one line. The lines that follow that line revert to being indented as specified by the ".in" that was in effect before the ".ti". This I-D uses ".ti" requests only to create hanging indents for bulleted and lettered paragraphs and for entries in Section 6, "References".

The nroff request ".ce N", where N is an integer, centers the next N input lines within the currently available horizontal output space, i.e., within the length (72 spaces) specified by the ".ll 7.2i" request in Fragment 1 minus the spaces for the ".in N" that is currently in effect.

The nroff request ".bp" terminates the current output page and begins a new page. In this I-D, this action causes the NP macro to execute as described in Section 4.3, placing a footer on the old page and a header on the new page. In a large or formal ISPD, the author MAY place a ".bp" before the level-one heading of each major section, so that these sections begin on a new page. However, that style is not often seen in RFCs.

The nroff input file for this I-D has ".bp" requests in the following places:

A. Before selected level-one headings:
   - "Table of Contents"
   - "Table of Figures"
   - "1. Introduction"
   - "2. Format Requirements for RFCs"
   - "3. The Word View"
   - "4. The nroff View"
   - "5. The Five Phases Step-by-Step"
   - "6. References"

B. On the last line of the file, to cause the NP macro to place a footer on the last page, as described for Fragment 4 in Section 4.3.2.

4.5 Pagination Enhancements

Word has a feature that can be used to prevent widows and orphans. (A "widow" is the last line of a paragraph printed by itself at

the top of a page, and an "orphan" is the first line of a paragraph printed by itself at the bottom of a page.) If a widow is about to occur on page, Word prints one less line of text on the previous page; if an orphan is about to occur, Word moves the line to the next page. However, this treatment is lost when a Word file is saved as a plain text file.

RFC 2223 requires neither control of widows and orphans nor other pagination enhancements. For simplicity, this I-D incorporates only a few enhancements, and all are made with the nroff request ".ne", as illustrated by Figure 5.

Figure 5. Examples of Keeping Lines Together

```
.ne 3    \" Keep the next three lines together on the same page.
. Heading at Level One
[One blank line here.]
.in 3
[One blank line here.]
   Paragraph at level one.
[One blank line here.]
.ne 3    \" Keep the next three lines together on the same page.
   1.1 Heading at Level Two
.in 6
[One blank line here.]
      Paragraph at level two.
[One blank line here.]
.ne 6    \" Keep the next six lines together on the same page.
      - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.ce 1
                  Exhibit N.  Table of Forty Integers
[One blank line here.]
       0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
      20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
      - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
[One blank line here.]
```

The nroff request ".ne N" indicates that N more lines are needed on the current output page. If they are not available, nroff terminates that output page and begins a new one. In this I-D, this causes the NP macro to execute as described in Section 4.3, placing a footer on the old page and a header on the new page.

The enhancements that have been made in this I-D, and are recommended for UGLI use, are as follows:

A. Each section heading is kept on the same page with at least one line of text from the paragraph that follows. This is done by placing a ".ne 3" request before each heading. This indicates the need for three more lines on the current output page: the

heading line, the blank line that follows the heading, and the first line of the paragraph.

B. All lines of an exhibit are kept on the same page. This is done with a ".ne N" request before each exhibit, where N is the number of lines in the exhibit, including the top and bottom delimiter lines.

C. Any item in Section 6, "References", that has only two lines is kept together on the same page by using ".ne 2" (although this turned out not be necessary in this I-D, as seen on page 35).

D. All the lines of Section 8, "Author's Address", are kept together on one page (although this turned out not be necessary in this I-D, as seen on page 35).

Enhancements "A" and "B" are equivalent to using Word's "keep together" feature. So the nroff output does not differ much from what is seen with Word's WYSIWYG editor. But widows and orphans can still occur in the nroff output. More enhancements can be made with ".ne" requests, but they produce diminishing returns because the nroff output begins to diverge from what is seen in Word, giving up some of the "dual-view" advantage of UGLI. For example, changing "A" to use ".ne 4" and putting ".ne 2" in front of each of the other paragraphs eliminates orphans, but still leaves ugly widows like the one on page 22. More work is needed to achieve the same pagination as Word's "widow and orphan control".

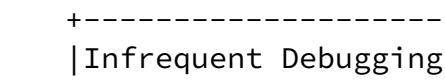
It is possible to construct nroff macros that control widows and orphans like Word does and make other, more sophisticated enhancements, too. For example, in the nroff input file for this I-D, an ".ne 10" request before Figure 1 causes extra, unwanted blank lines at the bottom of page 6. Trying to eliminate such gaps by manually rearranging lines of text in the nroff input file becomes extremely laborious, but a nroff macro in place of ".ne" could automatically "float" exhibits over text and keep pages full. Your UNIX environment may already have available sets of macros that are useful for producing RFCs, and such macros may be used in Phase 2.

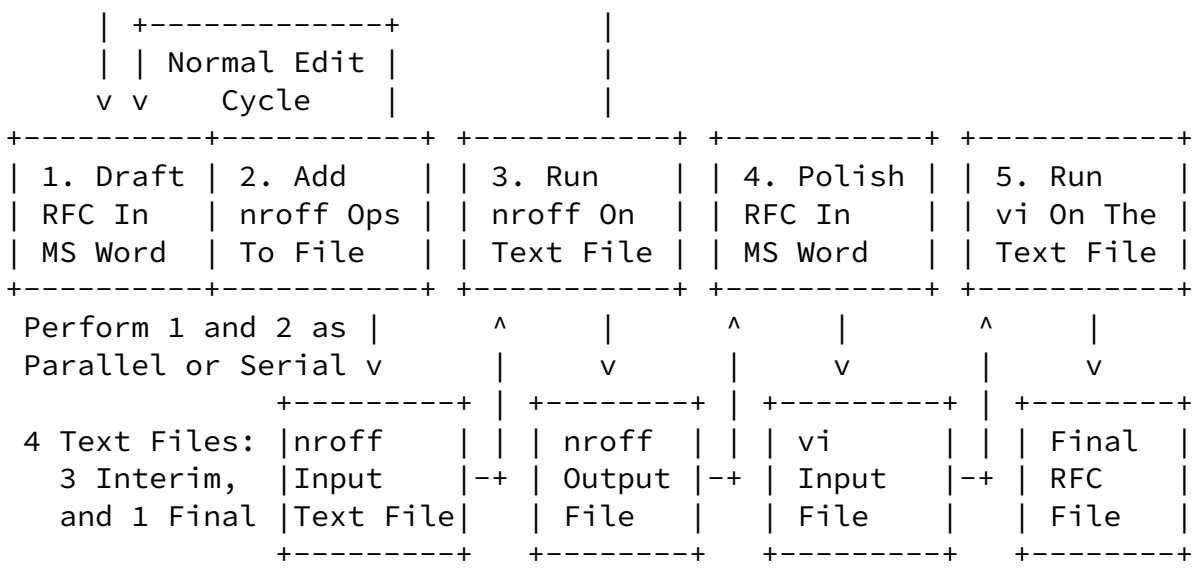
For another example, nroff outputs the last non-blank line of the last paragraph on page 20 immediately before the two blank lines of the footer. Therefore, the blank line that follows the paragraph in the nroff input file is output at the top of page 21, immediately after the two blank lines of the header, making the header look like it has three blank lines. A macro could eliminate such unwanted lines.

5. The Five Phases Step-by-Step

As shown in Figure 6, UGLI has five phases and uses three intermediate files. This section describes all five phases. Most of the details for Phases 1 and 2 have already been covered in Sections 2, 3, and 4. This section mainly describes how to move files between your Word environment and your UNIX environment, how to execute the nroff command in Phase 3, how to do final editing in Phase 4, and how to perform the vi operations in Phase 5. (You do not need previous knowledge of vi; this section provides complete instructions for the few operations that must be performed.)

Figure 6. UGLI Phases and Files





5.1 Draft Document in Word

In Phase 1, you use Word to compose your document in the RFC format described in Section 2. As described in Section 3, you can configure the Word file to closely resemble the appearance of the RFC format. This enables you to see approximately what the final product of Phase 5 will look like while you are still working in Word in Phases 1 and 2.

5.2 Prepare File for nroff Processing

In Phase 2, you create an input file for nroff processing in Phase 3. This includes adding nroff requests and properly saving the result as text file.

5.2.1 Create a Dual-View File

Section 4 describes how to enhance your Word document with nroff requests to create RFC formatting after the Word file is saved as text. The automatic formatting you established in Phase 1 provides a visual guide for inserting the requests correctly. (If you have access to a package of nroff macros

that can enhance pagination and perform other functions found in modern word processors, you can insert those macro calls in this phase.)

You achieve a "dual view" by formatting nroff requests as "hidden text" in the Word document. Then, you can turn off the hidden text in Word whenever you want to view or print a rough draft directly from Word without going through nroff.

5.2.2 Properly Save As a Text File

Next, save the Word file using the "Text Only with Line Breaks" option, not the "Text Only" option. In this discussion, we call the saved file "nroffin", because we use it as nroff's input file for Phase 3.

The distinction between "Text Only with Line Breaks" and "Text Only" is important. In Word, the former produces a text file in which every line that is displayed or printed from the original Word file results in a separate "line" in the text file, i.e., in a separate character string with its own CR and LF characters at the end.

For example, if we view this paragraph of the original Word file on my screen, we see 13 lines with a "paragraph mark" at the end of only the 13-th line. If we improperly save the original Word file as "Text Only", then this multi-line paragraph in the Word file results in a single "line" in the nroff input file (i.e., a string of characters terminated by one CR-LF). So, if we view the improperly saved paragraph in Word, it looks just like it did in the Word file; it has 13 lines and a paragraph mark at the end of the last line. If we properly save the file as "Text Only with Line Breaks", the result is 13 "lines" in the nroff input file. That is, we still see 13 lines when we view the file in Word, but now each line ends with a paragraph mark.

This difference affects nroff requests, such as ".in", which deal with "lines". When nroff processes the properly saved paragraph, it indents by adding nine blank characters at the front of each of the 13 "lines" it receives. When nroff processes the improperly saved paragraph, it adds nine characters at the front of the single "line" it receives; so that when we view the paragraph in the nroff output file, only the first line of the 13 is indented.

5.3 Process File in nroff

In Phase 2, you move the "nroffin" file to a UNIX system, process it with nroff to create an "nroffout" file, and move "nroffout" back to where you can work on it some more with Word.

The UNIX host that this author uses for UGLI is his departmental POP3 host, the one from which Eudora retrieves this author's mail. The user identifier and password that Eudora uses to retrieve mail provides a handy account for FTP and TELNET connections and for running nroff and vi. FTP and TELNET are provided on Macs by the freeware NCSA Telnet application. Windows users all have FTP and TELNET available in DOS or may instead use some Windows application that provides those protocols.

5.3.1 Move nroff Input to UNIX System

There are many ways to move "nroffin" to a UNIX system. If the machine on which you run Word is not connected to a computer network, you probably will use sneaker-net and a floppy disk. But if you have network connectivity and can arrange a login account on a UNIX host, your work will go much faster.

Run the NCSA Telnet program (or its equivalent on your platform) and "open" a connection to the UNIX host, checking the box that says "FTP session" (i.e., open a connection to port 21 on the host). This initiates a session in which your local computer is an FTP client, and the UNIX host is the FTP server. When the host says "ready", reply with "user". When the host says "Username:", reply with yours. When the host says "password required", reply with yours. When the host says "logged in", say "put nroffin". (Your actual dialog may differ slightly, depending on how your FTP client and server are implemented.)

This description omits some details and glosses over occasional frustrations. For example, the FTP client usually requires "nroffin" to be at the top level of the folder (i.e., directory) that is the hard disk on which the TELNET application resides, and that also is where "nroffout" is placed after being retrieved from the server. NCSA Telnet is touchy and sometimes fails unless a certain ritual is followed. Before logging in, look in your local folder for any old "nroffin" file and drag it to the Trash. Then, drag the new "nroffin" file from your desktop into that folder. After logging in but before saying "put", remove any old "nroffin"

from the UNIX host. Do this by first saying "ls", which is a UNIX command supported in NCSA Telnet to list files in the UNIX account. If there already is an "nroffin" on the UNIX host, say "rm nroffin", i.e., "remove" the old file.

Sometimes NCSA Telnet simply refuses to function and, in response to a "put", complains ("Cannot open file") no matter what is tried. In such cases, rebooting seems to be the cure.

5.3.2 Run nroff Command

In a separate window, use NCSA Telnet to "open" a TELNET connection to the same UNIX host (i.e., do not check the FTP box this time). When the host says "Login:", reply with your username again. When the host says "Password:", reply with yours again. When the host prompts for a command, say "nroff <nroffin >nroffout"; i.e., run nroff and specify "nroffin" as the input file and "nroffout" as the output file.

5.3.3 Get nroff Output from UNIX System

Finally, return to the first window, where you have the FTP session, and say "get nroffout". Now you have the nroff output file back on your local machine, ready for final editing in Phase 4.

5.4 Do Final Editing and Prepare for vi

In Phase 4, you open "nroffout" with Word and review the result of the nroff processing. If there are errors to correct or additions to make, return to Phase 1 or Phase 2. If not, do the following four steps to polish the draft and get ready for Phase 5:

A. Complete the content tables: Now that you can see which page each heading and exhibit is on, you write page numbers in the Table of Contents and in any other content tables you have.

B. (Optional) Add a line to page 1: If you wish, you can correct the length discrepancy of page 1 that is described in Section 2.1.1. To make all pages display and print identically, add a blank line to the beginning of the file, i.e., as the first line of page 1.

C. Insert markers for vi: In this author's environment, when "nroffout" is viewed in Word, lines equivalent to the four shown in Fragment 5 appear at each boundary between pages:

Fragment 5. Page Boundary in nroff Output File

```
<non-blank line of footer through page number>]<paragraph-mark>
<paragraph-mark>
<paragraph-mark>
Internet-Draft        UGLI        December 1999<paragraph-mark>
```

The first line is the last line of a page: a non-blank footer

line (which always ends with the "]" character after the page number) and its paragraph mark. The next two lines are empty. The fourth line is the first line of a page: the non-blank running header line. We would like to simply replace the second and third lines by a single line with <page break><paragraph-mark> and save the file as text, achieving the effect shown in Fragment 6 (where "FF", "CR", and "LF" represents the ASCII control characters), which would satisfy Rule 3:

Fragment 6. Page Boundary in RFC Format

```
<non-blank line of footer through page number>]<CR><LF>
<FF><CR><LF>
Internet-Draft        UGLI        December 1999<CR><LF>>
```

Unfortunately, Word discards page breaks when it saves a file as text. But vi does not discard FF characters when it saves a file. Therefore, we accomplish the change in two steps, one with Word and one with vi. First, we use Word to delete the second line and insert "QQQQ" in the third line as a marker for vi. To do this, use Word's "Replace All" option in "Find and Replace", replacing character strings as shown Figure 7.

Figure 7. "Find" and "Replace" Strings

```
 Before: ]<paragraph-mark><paragraph-mark><paragraph-mark>Int

 After:  ]<paragraph-mark>QQQQ<paragraph-mark>Int
```

Now when we view the file in Word, we see the following at each page boundary:

Fragment 7. Page Boundary in vi Input File

```
 <non-blank line of footer through page number>]<paragraph-mark>
 QQQQ<paragraph-mark>
 Internet-Draft        UGLI        December 1999<paragraph-mark>
```

The choice of "QQQQ" is not significant to either Word or vi. You can use any character string that is not used elsewhere in the file, such as "DSSCS". In the next phase, we will replace each "QQQQ" string with an ASCII FF control character.

(For this I-D, the character string "QQQQ" *does* appear in the text, so it is not usable for the substitution marker. Instead, this author uses the string ... Oops, sorry! If he tells you, it will be usable.)

D. Remove debris: In this author's environment, "nroffout" usually has some unwanted lines and fragmentary pages after the last real page. You use Word to delete everything that follows the "QQQQ<paragraph-mark>" on the last real page of your RFC.

After these editing steps, you save the Word file as text. This time, it does not matter whether you save as "Text Only" or "Text Only with Line Breaks", because every line you see in Word already has a paragraph mark at its end. We call the saved file "viin", because we use it as vi's input file for Phase 5.

5.5 Process File in vi

In Phase 5, using the same technique you used in Phase 3, you move the "viin" file to a host that has vi or a similar line editor

program. In UNIX, you then say "vi viin" to invoke the editor and open the file, and you type three commands.

A. At vi's prompt, you type ":1,$ s/QQQQ/<control-L>/". That is, between the second and third slashes, you enter the ASCII FF control character by striking the "L" key while holding down the "control" key. This string says, "This (:) is a vi command. From the first line of the file (1) to the last line ($), wherever there is the string 'QQQQ', substitute (s) an ASCII form feed character."

B. Next, type the command ":w viout", which says, "Write the file to a new file with the name 'viout'."

C. Finally, type the command ":q" to quit (i.e., exit, terminate) the vi program, and then move "viout" back to your local machine.

Congratulations! You now have a finished paper that satisfies the requirements of RFC 2223 ... and then some.

6. References

[ANSI] American National Standards Institute, "Information Systems --Coded Character Sets--7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)", ANSI X3.4-1986 (R1997).

[Hain] T. Hain, Using Microsoft Word to create Internet Drafts and RFC's, draft-hain-msword-template-00.txt, February 1999.

[Ossa] J. F. Ossanna and B. W. Kernighan, "Troff User's Manual", AT&T Bell Laboratories Murray Hill, NJ 07974, Computing Science Technical Report No. 54, revised Nov 1992.

[R854] J. Postel, "TELNET Protocol Specification", RFC 854, May 1983.

[R1208] O. Jacobsen and D. Lynch, "A Glossary of Networking Terms", RFC 1208, Mar 1991.

[R1983] G. Malkin, ed., "Internet Users' Glossary", FYI 18, RFC 1983, Aug 1996.

[R2026] S. Bradner, "The Internet Standards Process -- Revision 3", RFC 2026, BCP 9, Oct 1996.

[R2119] S. Bradner, "Key Words for Use in RFCs To Indicate Requirement Levels", BCP 14, RFC 2119, Mar 1997.

[R2223] J. Postel and J. Reynolds, "Instructions to RFC Authors", Oct 1997.

[Shir] R. Shirey, "Internet Security Glossary", <draft-shirey-security-glossary-01.txt>, 17 Oct 1999.

7. Security Considerations

This paper does not discuss security issues and has no security implications of which the author is aware.

8. Author's Address

Please address all comments to:

Robert W. Shirey
Email: rshirey@bbn.com
Phone: +1 (703) 284-4641
Fax: +1 (703) 284-2766

GTE / BBN Technologies
Suite 1200, Mail Stop 30/12B2
1300 Seventeenth Street North
Arlington, VA 22209-3801 USA

9. Expiration Date

This Internet Draft expires on 27 June 2000.