Microsoft® MS-DOS®, Windows®, OS/2®, and Apple® Macintosh® Applications

Version: RTF Version 1.3

Subject: **Rich Text Format (RTF) Specification and Sample RTF Reader Program**

Contents: 92 Pages, 1 Disk

Microsoft®

Product Support Services

Application Note

1/95 - GC0165

Table of Contents

INTRODUCTION

The Rich Text Format (RTF) Specification is a method of encoding formatted text and graphics for easy transfer between applications. Currently, users depend on special translation software to move word-processing documents between different MS-DOS, Windows, OS/2, and Apple Macintosh applications.

The RTF Specification provides a format for text and graphics interchange that can be used with different output devices, operating environments, and operating systems. RTF uses the ANSI, PC-8, Macintosh, or IBM® PC character set to control the representation and formatting of a document, both on the screen and in print. With the RTF Specification, documents created under different operating systems and with different software applications can be transferred between those operating systems and applications.

Software that takes a formatted file and turns it into an RTF file is called a *writer*. An RTF writer separates the application's control information from the actual text and writes a new file containing the text and the RTF groups associated with that text. Software that translates an RTF file into a formatted file is called a *reader*.

Included with the RTF specification is a sample RTF reader application (see "Appendix A: Sample RTF Reader Application" beginning on page 55). It is designed for use with the specification to assist those users developing their own RTF readers. The GC0165 disk included with this Application Note contains RTFREADR.EXE, the sample RTF reader program itself. This file and its use are described in Appendix A. The sample RTF reader is not a for-sale product, and Microsoft does not provide technical or any other type of support for the sample RTF reader code or the RTF specification.

RTF SYNTAX

An RTF file consists of unformatted text, control words, control symbols, and groups. For ease of transport, a standard RTF file can consist of only 7-bit ASCII characters. (Converters that communicate with Microsoft Word for Windows or Microsoft Word for the Macintosh should expect 8-bit characters.)

A *control word* is a specially formatted command that RTF uses to mark printer control codes and information that applications use to manage documents. A control word takes the following form:

\ LetterSequence<Delimiter>

Note that a backslash begins each control word.

The LetterSequence is made up of lowercase alphabetic characters between "a" and "z" inclusive. RTF is case sensitive, and all RTF control words must be in lowercase.

The Delimiter marks the end of an RTF control word, and can be one of the following:

- A space. In this case, the space is part of the control word.
- A digit or a hyphen (-), which indicates that a numeric parameter follows. The subsequent digital sequence is then delimited by a space or any character other than a letter or a digit. The parameter can be a positive or a negative number. The range of the values for the number is –32767 through 32767. However, Microsoft Word for Windows, Word for OS/2, and Word for the Macintosh restrict the range to –31680 through 31680. If a numeric parameter immediately follows the control word, this parameter becomes part of the control word. The control word is then delimited by a space or a nonalphabetic or nonnumeric character in the same manner as any other control word.
- Any character other than a letter or a digit. In this case, the delimiting character terminates the control word but is not actually part of the control word.

If a space delimits the control word, the space does not appear in the document. Any characters following the delimiter, including spaces, will appear in the document. For this reason, you should use spaces only where necessary; do not use spaces merely to break up RTF code.

A *control symbol* consists of a backslash followed by a single, nonalphabetic character. For example, **\~** represents a nonbreaking space. Control symbols take no delimiters.

A *group* consists of text and control words or control symbols enclosed in braces ({ }). The opening brace ({) indicates the start of the group and the closing brace (}) indicates the end of the group. Each group specifies the text affected by the group and the different attributes of that text. The RTF file can also include groups for fonts,

styles, screen color, pictures, footnotes, annotations, headers and footers, summary information, fields, and bookmarks, as well as document-, section-, paragraph-, and character-formatting properties. If the font, file, style, screen-color, revision mark, and summary-information groups and document-formatting properties are included, they must precede the first plain-text character in the document. These groups form the RTF file header. If the group for fonts is included, it should precede the group for styles. If any group is not used, it can be omitted. The groups are discussed in the following sections.

The control properties of certain control words (such as bold, italic, keep together, and so forth) have only two states. When such a control word has no parameter or has a nonzero parameter, it is assumed that the control word turns on the property. When such a control word has a parameter of 0 (zero), it is assumed that the control word turns off the property. For example, **b** turns on bold, whereas **b0** turns off bold.

Certain control words, referred to as *destinations*, mark the beginning of a collection of related text that could appear at another position, or destination, within the document. Destinations may also be text that is used but should not appear within the document at all. An example of a destination is the **footnote** group, where the footnote text follows the control word. Destination control words and their following text must be enclosed in braces. Destinations added after the RTF Specification published in the March 1987 *Microsoft Systems Journal* may be preceded by the control symbol **. This control symbol identifies destinations whose related text should be ignored if the RTF reader does not recognize the destination. (RTF writers should follow the convention of using this control symbol when adding new destinations or groups.) Destinations whose related text should be inserted into the document even if the RTF reader does not recognize the destination should not use *. All destinations that were not included in the March 1987 revision of the RTF Specification are shown with * as part of the control word.

Formatting specified within a group affects only the text within that group. Generally, text within a group inherits the formatting of the text in the preceding group. However, Microsoft implementations of RTF assume that the footnote, annotation, header, and footer groups (described later in this chapter) do not inherit the formatting of the preceding text. Therefore, to ensure that these groups are always formatted correctly, you should set the formatting within these groups to the default with the **sectd**, **pard**, and **plain** control words, and then add any desired formatting.

The control words, control symbols, and braces constitute control information. All other characters in the file are plain text. Here is an example of plain text that does not exist within a group:

```
{\rtf\ansi\deff0{\fonttbl{\f0\froman Tms Rmn;}{\f1\fdecor
Symbol;}{\f2\fswiss Helv;}}{\colortbl;\red0\green0\blue0;
\red0\green0\blue255;\red0\green255\blue255;\red0\green255\
blue0;\red255\green0\blue255;\red255\green0\blue0;\red255\
green255\blue0;\red255\green255\blue255;}{\stylesheet{\fs20 \snext0Normal;}}{\info{\author John Doe}
{\creatim\yr1990\mo7\dy30\hr10\min48}{\version1}{\edmins0}
{\nofpages1}{\nofwords0}{\nofchars0}{\vern8351}}\widoctrl\ftnbj \sectd\linex0\endnhere \pard\plain \
fs20 This is plain text.\par}
```

The phrase “This is plain text” is not part of a group and is treated as document text.

As previously mentioned, the backslash (\) and braces ({ }) have special meaning in RTF. To use these characters as text, precede them with a backslash, as in \\, \{, and \}.

CONVENTIONS OF AN RTF READER

The reader of an RTF stream is concerned with the following:

- Separating control information from plain text.
- Acting on control information.
- Collecting and properly inserting text into the document, as directed by the current group state.

Acting on control information is designed to be a relatively simple process. Some control information simply contributes special characters to the plain text stream. Other information serves to change the *program state*, which includes properties of the document as a whole, or to change any of a collection of *group states*, which apply to parts of the document.

As previously mentioned, a group state can specify the following:

- The *destination*, or part of the document that the plain text is constructing.
- Character-formatting properties, such as bold or italic.
- Paragraph-formatting properties, such as justified or centered.
- Section-formatting properties, such as the number of columns.
- Table-formatting properties, which define the number of cells and dimensions of a table row.

In practice, an RTF reader will evaluate each character it reads in sequence as follows:

- If the character is an opening brace ({), the reader stores its current state on the stack. If the character is a closing brace (}), the reader retrieves the current state from the stack.
- If the character is a backslash (\), the reader collects the control word or control symbol and its parameter, if any, and looks up the control word or control symbol in a table that maps control words to actions. It then carries out the action prescribed in the table. (The possible actions are discussed below.) The read pointer is left before or after a control-word delimiter, as appropriate.
- If the character is anything other than an opening brace ({), closing brace (}), or backslash (\) , the reader assumes that the character is plain text and writes the character to the current destination using the current formatting properties.

If the RTF reader cannot find a particular control word or control symbol in the look-up table described above, the control word or control symbol should be ignored. If a control word or control symbol is preceded by an opening brace ({), it is part of a group. The current state should be saved on the stack, but no state change should occur. When a closing brace (}) is encountered, the current state should be retrieved from the stack, thereby resetting the current state. If the * control symbol precedes a control word, then it defines a destination group and was itself preceded by an opening brace ({). The RTF reader should discard all text up to and including the closing brace (}) that closes this group. All RTF readers must recognize all destinations defined in the March 1987 RTF Specification. The reader may skip past the group, but it is not allowed to simply discard the control word. Destinations defined since March 1987 are marked with the * control symbol.

> ***Note:*** *All RTF readers must implement the * control symbol so that they can read RTF files written by newer RTF writers.*

For control words or control symbols that the RTF reader can find in the look-up table, the possible actions are as follows.

Action	Description
Change Destination	The RTF reader changes the destination to the destination described in the table entry. Destination changes are legal only immediately after an opening brace ({). (Other restrictions may also apply; for example, footnotes cannot be nested.) Many destination changes imply that the current property settings will be reset to their default settings. Examples of control words that change destination are **\footnote**, **\header**, **\footer**, **\pict**, **\info**, **\fonttbl**, **\stylesheet**, and **\colortbl**. This Application Note identifies all destination control words where they appear in control-word tables.
Change Formatting Property	The RTF reader changes the property as described in the table entry. The entry will specify whether a parameter is required. The “APPENDIX C: INDEX OF RTF CONTROL WORDS” section at the end of this Application Note also specifies which control words require parameters. If a parameter is needed and not specified, then a default value will be used. The default value used depends on the control word. If the control word does not specify a default, then all RTF readers should assume a default of 0.
Insert Special Character	The reader inserts into the document the character code or codes described in the table entry.

Insert Special Character and Perform Action	The reader inserts into the document the character code or codes described in the table entry and performs whatever other action the entry specifies. For example, when Microsoft Word interprets **\par**, a paragraph mark is inserted in the document and special code is run to record the paragraph properties belonging to that paragraph mark.

FORMAL SYNTAX

This Application Note describes RTF using the following syntax, based on Backus-Naur Form.

Syntax	Meaning
#PCDATA	Text (without control words).
#SDATA	Hexadecimal data.
#BDATA	Binary data.
'c'	A literal.
<text>	A nonterminal.
a	The (terminal) control word a, without a parameter.
a or **a*N***	The (terminal) control word a, with a parameter.
a?	Item a is optional.
a+	One or more repetitions of item a.
a*	Zero or more repetitions of item a.
a b	Item a followed by item b.
a | b	Item a or item b.
a & b	Item a and/or item b, in any order.

CONTENTS OF AN RTF FILE

An RTF file has the following syntax:

<File> '{' <header> <document>'}'

This syntax is the standard RTF syntax; any RTF reader must be able to correctly interpret RTF written to this syntax. It is worth mentioning again that RTF readers do not have to use all control words, but they must be able to harmlessly ignore unknown (or unused) control words, and they must correctly skip over destinations marked with the ***** control symbol. There may, however, be RTF writers that generate RTF that does not conform to this syntax, and as such, RTF readers should be robust enough to handle some minor variations. Nonetheless, if an RTF writer generates RTF conforming to this specification, then any correct RTF reader should be able to interpret it.

Header

The header has the following syntax:

<header> ***rtf*** <charset> ***deff***? <fonttbl> <filetbl>? <colortbl>? <stylesheet>? <revtbl>?

RTF Version

An entire RTF file is considered a group and must be enclosed in braces. The control word **rtf*N must follow the opening brace. The numeric parameter ***N*** identifies the major version of the RTF Specification used. The RTF standard described in this Application Note, although titled as version 1.2, continues to correspond syntactically to RTF Specification Version 1. Therefore, the numeric parameter ***N*** for the **rtf** control word should still be emitted as 1.

Character Set

After specifying the RTF version, you must declare the character set used in this document. The control word for the character set must precede any plain text or any table control words. The RTF Specification currently supports the following character sets.

Control word	Character set
\ansi	ANSI (the default)
\mac	Apple Macintosh
\pc	IBM PC code page 437
\pca	IBM PC code page 850, used by IBM Personal System/2® (not implemented in version 1 of Microsoft Word for OS/2)

Font Table

The **fonttbl** control word introduces the font table group. Unique **\f*N*** control words define each font available in the document, and are used to reference that font throughout the document. This group has the following syntax:

<fonttbl>	'{' **\fonttbl** (<fontinfo> | ('{' <fontinfo> '}'))+ '}'
<fontinfo>	<fontnum><fontfamily><fcharset>?<fprq>?<fontemb>?<codepage>? <fontname><fontaltname>? ';'
<fontnum>	***f***
<fontfamily>	**\fnil** | **\froman** | **\fswiss** | **\fmodern** | **\fscript** | **\fdecor** | **\ftech** | **\fbidi**
<fcharset>	***fcharset***
<fprq>	***fprq***
<fontname>	#PCDATA
<fontaltname>	'{*' **\falt** #PCDATA '}'
<fontemb>	'{*' **\fontemb** <fonttype> <fontfname>? <data>? '}'
<fonttype>	**\ftnil** | **\fttruetype**
<fontfname>	'{*' **\fontfile** <codepage>? #PCDATA '}'
<codepage>	\cpg

Note for <fontemb> that either <fontfname> or <data> must be present, although both may be present.

All fonts available to the RTF writer can be included in the font table, even if the document doesn't use all the fonts.

RTF also supports font families, so that applications can attempt to intelligently choose fonts if the exact font is not present on the reading system. RTF uses the following control words to describe the various font families.

Control word	Font family	Examples

\fnil	Unknown or default fonts (the default)	
\froman	Roman, proportionally spaced serif fonts	Times New Roman®, Palatino®
\fswiss	Swiss, proportionally spaced sans serif fonts	Arial®
\fmodern	Fixed-pitch serif and sans serif fonts	Courier New®, Pica
\fscript	Script fonts	Cursive
\fdecor	Decorative fonts	Old English, ITC Zapf Chancery®
\ftech	Technical, symbol, and mathematical fonts	Symbol
\fbidi	Arabic, Hebrew, or other bidirectional font	Miriam

If an RTF file uses a default font, the default font number is specified with the \ **deff*N*** control word, which must precede the font-table group. The RTF writer supplies the default font number used in the creation of the document as the numeric argument ***N***. The RTF reader then translates this number through the font table into the most similar font available on the reader's system.

The following control words specify the character set, alternative font name and pitch of a font in the font table.

Control word	Definition
\fcharset*N*	Specifies the character set of a font in the font table.
\falt	Indicates alternate font name to use if the specified font in the font table is not available. **'{*' \falt** <Alternate Font Name>**'}'**
\fprq*N*	Specifies the pitch of a font in the font table.

If **\fcharset** is specified, the ***N*** argument can be one of the following types.

Character set	***N* Value**
ANSI_CHARSET	0
SYMBOL_CHARSET	2
SHIFTJIS_CHARSET	128
GREEK_CHARSET	161
TURKISH_CHARSET	162
HEBREW_CHARSET	177
ARABICSIMPLIFIED_CHARSET	178
ARABICTRADITIONAL_CHARSET	179
ARABICUSER_CHARSET	180
HEBREWUSER_CHARSET	181
CYRILLIC_CHARSET	204
EASTERNEUROPE_CHARSET	238
PC437_CHARSET	254
OEM_CHARSET	255

If **\fprq** is specified, the ***N*** argument can be one of the following values.

Pitch	Value
Default pitch	0
Fixed pitch	1
Variable pitch	2

Font Embedding

RTF supports embedded fonts with the **fontemb** group located inside a font definition. An embedded font can be specified by a filename, or the actual font data may be located inside the group. If a filename is specified, it is contained in the **fontfile** group. The **cpg** control word can be used to specify the character set for the filename.

RTF supports TrueType® and other embedded fonts. The type of the embedded font is described by the following control words.

Control word	Embedded font type
\ftnil	Unknown or default font type (the default)
\fttruetype	TrueType font

Code Page Support

A font may have a different character set from the character set of the document. For example, the Symbol font has the same characters in the same positions both on the Macintosh and in Windows. RTF describes this with the **cpg** control word, which names the character set used by the font. In addition, filenames (used in field instructions and in embedded fonts) may not necessarily be the same as the character set of the document; the **cpg** control word can change the character set for these filenames as well. However, all RTF documents must still declare a character set (that is, **\ansi, \mac, \pc,** or **\pca**) to maintain backwards compatibility with earlier RTF readers.

The table below describes valid values for **cpg**.

Value	Description
437	United States IBM
708	Arabic (ASMO 708)
709	Arabic (ASMO 449+, BCON V4)
710	Arabic (transparent Arabic)
711	Arabic (Nafitha Enhanced)
720	Arabic (transparent ASMO)
819	Windows 3.1 (United States and Western Europe)
850	IBM multilingual
852	Eastern European
860	Portuguese
862	Hebrew
863	French Canadian
864	Arabic
865	Norwegian
866	Soviet Union
932	Japanese
1250	Windows 3.1 (Eastern European)
1251	Windows 3.1 (Cyrillic)

File Table

The **\filetbl** control word introduces the file table destination. This group defines the files referenced in the document and has the following syntax:

<filetbl>	'{*' **\filetbl** ('{' <fileinfo> '}')+ '}'
<fileinfo>	**file** <filenum><relpath>?<osnum>? <filesource>+ <filename>
<filenum>	***fid***
<relpath>	***frelative***
<osnum>	***fosnum***
<filesource>	**\fvalidmac** | **\fvaliddos** | **\fvalidntfs** | **\fvalidhpfs** | **\fnetwork**
<filename>	#PCDATA

Note that the filename can be any valid alphanumeric string for the named file system, indicating the complete path and filename.

Control word	Definition
\filetbl	A list of documents referenced by the current document. The file table has a structure analogous to the style or font table. This is a destination control word output as part of the document header.
\file	Marks the beginning of a file group, which lists relevant information about the referenced file. This is a destination control word.
\fid*N*	File ID number. Files are referenced later in the document using this number.
\frelative*N*	The character position within the path (starting at 0 [zero]) where the referenced file's path starts to be relative to the path of the owning document. For example, a document is saved to the path C:\PRIVATE\RESUME\FILE1.DOC and its file table contains the path C:\PRIVATE\RESUME\EDU\FILE2.DOC, then that entry in the file table will be **frelative**18, to point at the character "e" in "edu". This allows preservation of relative paths.
\fosnum*N*	Currently only filled in for paths from the Macintosh file system. It is an operating-system-specific number for identifying the file, which may be used to speed up access to the file, or find it if the file has been moved to another folder or disk. The Macintosh operating system name for this number is the "file id." Additional meanings of the **fosnum*N*** control word may be defined for other file systems in the future.
\fvalidmac	Macintosh file system.
\fvaliddos	MS-DOS file system.
\fvalidntfs	NTFS file system.
\fvalidhpfs	HPFS file system.
\fnetwork	Network file system. This control word may be used in conjunction with any of the previous file source control words.

Color Table

The **colortbl** control word introduces the color table group, which defines screen colors, character colors, and other color information. This group has the following syntax:

<colortbl>	'{' **\colortbl** <colordef>+ '}'
<colordef>	***red*** ? & ***green*** ? & ***blue*** ? ';'

The following are valid control words for this group.

Control word	Meaning
\red*N*	Red index
\green*N*	Green index
\blue*N*	Blue index

Each definition must be delimited by a semicolon, even if the definition is omitted. If a color definition is omitted, the RTF reader uses its default color. In the example below, three colors are defined. The first color is omitted, as shown by the semicolon following the **colortbl** control word.

```
{\colortbl;\red0\green0\blue0;\red0\green0\blue255;}
```

The foreground and background colors use indexes into the color table to define a color. For more information on color setup, see your Windows documentation.

The following example defines a block of text in color (where supported). Note that the **cf**/**cb** index is the index of an entry in the color table, which represents a red/green/blue color combination.

```
{\f1\cb1\cf2 This is colored text. The background is color
1 and the foreground is color 2.}
```

If the file is translated for software that does not display color, the reader ignores the color-table group.

Style Sheet

The **stylesheet** control word introduces the style sheet group, which contains definitions and descriptions of the various styles used in the document. All styles in the document's style sheet can be included, even if not all the styles are used. In RTF, a style is a form of shorthand used to specify a set of character, paragraph, or section formatting.

The style-sheet group has the following syntax:

<stylesheet>	'{' **\stylesheet** <style>+ '}'
<style>	'{' <styledef>?<keycode>? <formatting> <additive>? <based>? <next>? <stylename>? ';' '}'
<styledef>	***s*** | ***cs*** | ***ds***
<keycode>	'{' **\keycode** <keys> '}'
<additive>	**\additive**
<based>	***sbasedon***
<next>	***snext***
<formatting>	(<brdrdef> | <parfmt> | <apoctl> | <tabdef> | <shading> | <chrfmt>)+
<stylename>	#PCDATA
<keys>	(**\shift**? & **\ctrl**? & **\alt**?) <key>
<key>	***fn*** | #PCDATA

For <style>, both <styledef> and <stylename> are optional; the default is paragraph style 0. Note for <stylename> that Microsoft Word for the Macintosh interprets commas in #PCDATA as separating style synonyms. Also, for <key>, the data must be exactly one character.

Control word	Meaning
\cs*N*	Designates character style.

\s*N*	Designates paragraph style.
\ds*N*	Designates section style.
\additive	Used in a character style definition ('**{*\cs**···'}'). Indicates that style attributes are to be applied in addition to current attributes, rather than setting the character attributes to only the style definition.
\sbasedon*N*	Defines the number of the style on which the current style is based (the default is 222—no style).
\snext*N*	Defines the next style associated with the current style; if omitted, the next style is the current style.
\keycode	This group is specified within the description of a style in the style sheet in the RTF header. The syntax for this group is '**{*\keycode** <keys>**}**' where <*keys*> are the characters used in the key code. For example, a style, Normal, may be defined {\s0 {*\keycode \shift\ctrl n}Normal;} within the RTF style sheet. See the Special Character control words for the characters outside the alphanumeric range that may be used.
\alt	The ALT modifier key. Used to describe shortcut-key codes for styles.
\shift	The SHIFT modifier key. Used to describe shortcut-key codes for styles.
\ctrl	The CTRL modifier key. Used to describe shortcut-key codes for styles.
\fn*N*	Specifies a function key where ***N*** is the function key number. Used to describe shortcut-key codes for styles.

The following is an example of an RTF style sheet

```
{\stylesheet{\fs20 \sbasedon222\snext0{\*\keycode \shift\ctrl n}
Normal;}{\s1\qr \fs20 \sbasedon0\snext1 FLUSHRIGHT;}{\s2\fi-720\li720\fs20\ri2880\sbasedon0\snext2
IND;}}
```

and RTF paragraphs to which the styles are applied:

```
\widowctrl\ftnbj\ftnrestart \sectd \linex0\endnhere \pard\plain
\fs20 This is Normal style.
\par \pard\plain \s1\qr\fs20
This is right justified. I call this style FLUSHRIGHT.
\par \pard\plain \s2\fi-720\li720\fs20\ri2880
This is an indented paragraph. I call this style IND. It produces
a hanging indent.
\par}
```

Some of the control words in this example are discussed in later sections. In the example, note that the properties of the style were emitted following the application of the style. This was done for two reasons: 1) to allow RTF readers that don't support styles to still retain all formatting, and 2) to allow the additive model for styles, where additional property changes are "added" on top of the defined style. Some RTF readers may not "apply" a style upon only encountering the style number without the accompanying formatting information because of this.

Revision Marks

This table allows tracking of multiple authors and reviewers of a document, and is used in conjunction with the character properties for revision marks.

Control word	Definition

\revtbl This group consists of subgroups that each identify the author of a revision in the document, as in `{Author1;}`. This is a destination control word.

Revision conflicts, such as one author deleting another's additions, are stored as one group, in the following form:

CurrentAuthor\'00\'<length of previous author's name>PreviousAuthor\'00 PreviousRevisionTime

The 4 bytes of the Date/Time (DTTM) structure are emitted as ASCII characters, so values greater than 127 should be emitted as quoted hexadecimal values.

All time references for revision marks use the following bit field structure, DTTM.

Bit numbers	Information	Range
0–5	Minute	0–59
6–10	Hour	0–23
11–15	Day of month	1–31
16–19	Month	1–12
20–28	Year	= Year - 1900
29–31	Day of week	0 (Sun)–6 (Sat)

Document Area

Once the RTF header is defined, the RTF reader has enough information to correctly read the actual document text. The document area has the following syntax.

<document> <info>? <docfmt>* <section>+

Information Group

The **info** control word introduces the information group, which contains information about the document. This can include the title, author, keywords, comments, and other information specific to the file. This information is for use by a document-management utility, if available.

This group has the following syntax.

<info> '{' <title>? & <subject>? & <author>? & <operator>? & <keywords>? & <comment>? & ***version***? & <doccomm>? & ***vern***? & <creatim>? & <revtim>? & <printim>? & <buptim>? & ***edmins***? & ***nofpages***? & ***nofwords***? ***nofchars***? & ***id***? '}'

<title> '{' **title** #PCDATA '}'

<subject> '{' **subject** #PCDATA '}'

<author> '{' **author** #PCDATA '}'

<operator> '{' **operator** #PCDATA '}'

<keywords> '{' **keywords** #PCDATA '}'

<comment> '{' **comment** #PCDATA '}'

<doccomm> '{' **doccomm** #PCDATA '}'

<creatim> '{' **creatim** <time> '}'

<revtim>	'{' **\revtim** <time> '}'
<printim>	'{' **\printim** <time> '}'
<buptim>	'{' **\buptim** <time> '}'
<time>	***yr*** ? ***mo*** ? ***dy*** ? ***hr*** ? ***min*** ? ***sec*** ?

Some applications, such as Word, ask the user to type this information when saving the document in its native format. If the document is then saved as an RTF file or translated into RTF, the RTF writer specifies this information using the following control words. These control words are destinations and both the control words and the text should be enclosed in braces ({ }).

Control word	Meaning
\title	Title of the document. This is a destination control word.
\subject	Subject of the document. This is a destination control word.
\author	Author of the document. This is a destination control word.
\operator	Person who last made changes to the document. This is a destination control word.
\keywords	Selected keywords for the document. This is a destination control word.
\comment	Comments; text is ignored. This is a destination control word.
\version*N*	Version number of the document.
\doccomm	Comments displayed in Word's Edit Summary Info dialog box. This is a destination control word.

The RTF writer may automatically enter other control words, including the following.

Control word	Meaning
\vern*N*	Internal version number
\creatim	Creation time
\revtim	Revision time
\printim	Last print time
\buptim	Backup time
\edmins*N*	Total editing time (in minutes)
\yr*N*	Year
\mo*N*	Month
\dy*N*	Day
\hr*N*	Hour
\min*N*	Minute
\sec*N*	Seconds
\nofpages*N*	Number of pages
\nofwords*N*	Number of words
\nofchars*N*	Number of characters
\id*N*	Internal ID number

Any control word described in the previous table that does not have a numeric parameter specifies a date; all dates are specified with the **\yr \mo \dy \hr \min \sec** controls.

An example of an information group follows:

```
{\info{\title The Panda's Thumb}{\author Stephen J Gould}{\keywords
science natural history }}
```

Document Formatting Properties

After the information group (if any), there may be some document formatting control words (described as <docfmt> in the document area syntax description). These control words specify the attributes of the document, such as margins and footnote placement. These attributes must precede the first plain-text character in the document.

The control words that specify document formatting are listed in the following table (measurements are in twips; a twip is one-twentieth of a point). For omitted control words, RTF uses the default values.

Control word	Meaning
\deftab*N*	Default tab width in twips (the default is 720).
\hyphhotz*N*	Hyphenation hot zone in twips (the amount of space at the right margin in which words are hyphenated).
\hyphconsec*N*	***N*** is the maximum number of consecutive lines that will be allowed to end in a hyphen. 0 means no limit.
\hyphcaps	Toggles hyphenation of capitalized words (the default is on). Append 1 or leave control word by itself to toggle property on; append 0 (zero) to turn it off.
\hyphauto	Toggles automatic hyphenation (the default is off). Append 1 or leave control word by itself to toggle property on; append 0 (zero) to turn it off.
\linestart*N*	Beginning line number (the default is 1).
\fracwidth	Uses fractional character widths when printing (QuickDraw™ only).
***\nextfile**	Destination; the argument is the name of the file to print or index next; it must be enclosed in braces. This is a destination control word.
***\template**	Destination; the argument is the name of a related template file; it must be enclosed in braces. This is a destination control word.
\makebackup	Backup copy is made automatically when the document is saved.
\defformat	Tells the RTF reader that the document should be saved in RTF format.
\psover	Prints PostScript® over the text.
\doctemp	Document is a boilerplate document. For Word for Windows, this is a template; for Word for the Macintosh, this is a stationery file.
\deflang*N*	Defines the default language used in the document used with a **\plain** control word. See the section "Character-Formatting Properties" on page 33 of this Application Note for a list of possible values for ***N***.
Footnotes and Endnotes	
\fet*N*	Footnote/endnote type. This indicates what type of notes are present in the document. 0 Footnotes only or nothing at all (the default). 1 Endnotes only. 2 Footnotes and endnotes both. For backward compatibility, if **\fet1** is emitted, **\endnotes** or **\enddoc** will be emitted along with **aendnotes** or **aenddoc. RTF r**eaders that understand **\fet** will need to ignore the footnote-positioning control words, and use the endnote control words instead.
\ftnsep	Text argument separates footnotes from the document. This is a destination control word.

\ftnsepc	Text argument separates continued footnotes from the document. This is a destination control word.
\ftncn	Text argument is a notice for continued footnotes. This is a destination control word.
\aftnsep	Text argument separates endnotes from the document. This is a destination control word.
\aftnsepc	Text argument separates continued endnotes from the document. This is a destination control word.
\aftncn	Text argument is a notice for continued endnotes. This is a destination control word.
\endnotes	Footnotes at the end of the section (the default).
\enddoc	Footnotes at the end of the document.
\ftntj	Footnotes beneath text (top justified).
\ftnbj	Footnotes at the bottom of the page (bottom justified).
\aendnotes	Endnotes at end of section (the default).
\aenddoc	Endnotes at end of document.
\aftnbj	Endnotes at bottom of page (bottom justified).
\aftntj	Endnotes beneath text (top justified).
\ftnstart*N*	Beginning footnote number (the default is 1).
\aftnstart*N*	Beginning endnote number (the default is 1).
\ftnrstpg	Restart footnote numbering each page.
\ftnrestart	Footnote numbers restart at each section. Microsoft Word for the Macintosh uses this control to restart footnote numbering at each page.
\ftnrstcont	Continuous footnote numbering (the default).
\aftnrestart	Restart endnote numbering each section.
\aftnrstcont	Continuous endnote numbering (the default).
\ftnnar	Footnote numbering—Arabic numbering (1, 2, 3, ⋯)
\ftnnalc	Footnote numbering—Alphabetic lowercase (a, b, c, ⋯)
\ftnnauc	Footnote numbering—Alphabetic uppercase (A, B, C, ⋯)
\ftnnrlc	Footnote numbering—Roman lowercase (i, ii, iii, ⋯)
\ftnnruc	Footnote numbering—Roman uppercase (I, II, III, ⋯)
\ftnnchi	Footnote numbering—Chicago Manual of Style (*, †, ‡, §)
\aftnnar	Endnote numbering—Arabic numbering (1, 2, 3, ⋯)
\aftnnalc	Endnote numbering—Alphabetic lowercase (a, b, c, ⋯)
\aftnnauc	Endnote numbering—Alphabetic uppercase (A, B, C, ⋯)
\aftnnrlc	Endnote numbering—Roman lowercase (i, ii, iii, ⋯)
\aftnnruc	Endnote numbering—Roman uppercase (I, II, III, ⋯)
\aftnnchi	Endnote numbering—Chicago Manual of Style (*, †, ‡, §)

Page Information

\paperw*N*	Paper width in twips (the default is 12,240).
\paperh*N*	Paper height in twips (the default is 15,840).

\psz*N*	Used to differentiate between paper sizes with identical dimensions under Windows NT™. Values 1–41 correspond to paper sizes defined in DRIVINI.H in the Windows 3.1 SDK (DMPAPER_ values). Values greater than or equal to 42 correspond to user-defined forms under Windows NT.
\margl*N*	Left margin in twips (the default is 1800).
\margr*N*	Right margin in twips (the default is 1800).
\margt*N*	Top margin in twips (the default is 1440).
\margb*N*	Bottom margin in twips (the default is 1440).
\facingp	Facing pages (activates odd/even headers and gutters).
\gutter*N*	Gutter width in twips (the default is 0).
\margmirror	Switches margin definitions on left and right pages. Used in conjunction with **\facingp.**
\landscape	Landscape format.
\pgnstart*N*	Beginning page number (the default is 1).
\widowctrl	Enable widow and orphan control.

Linked Styles

\linkstyles	Update document styles automatically based on template.

Compatibility Options

\notabind	Don't add automatic tab stop for hanging indent.
\wraptrsp	Wrap trailing spaces onto the next line.
\prcolbl	Print all colors as black.
\noextrasprl	Don't add extra space to line height for showing raised/lowered characters.
\nocolbal	Don't balance columns.
\cvmme	Treat old-style escaped quotation marks (\") as current style ("") in mail merge data documents.
\sprstsp	Suppress extra line spacing at top of page. Basically, this means to ignore any line spacing larger than Auto at the top of a page.
\sprsspbf	Suppress space before paragraph property after hard page or column break.
\otblrul	Combine table borders as done in Word 5.x for the Macintosh. Contradictory table border information is resolved in favor of the first cell.
\transmf	Metafiles are considered transparent; don't blank the area behind metafiles.
\swpbdr	If a paragraph has a left border (not a box) and the Different Odd And Even or Mirror Margins check box is selected, Word will print the border on the right for odd-numbered pages.
\brkfrm	Show hard (manual) page breaks and column breaks in frames.

Forms

\formprot	This document is protected for forms.
\allprot	This document has no unprotected areas.
\formshade	This document has form field shading on.
\formdisp	This document currently has a forms drop down or check box selected.
\printdata	This document has print form data only on.

Revision Marks	
\revprot	This document is protected for revisions. The user can edit the document, but revision marking cannot be disabled.
\revisions	Turns on revision marking.
\revprop*N*	Argument indicates how revised text will be displayed: 0 for no properties shown; 1 for bold; 2 for italic; 3 for underline (the default); 4 for double underline.
\revbar*N*	Vertical lines mark altered text, based on the argument: 0 for no marking; 1 for left margin; 2 for right margin; 3 for outside (the default: left on left pages, right on right pages).
Annotations	
\annotprot	This document is protected for annotations. The user cannot edit the document but can insert annotations.
Bidirectional Controls	
\rtldoc	This document will be formatted to have Arabic-style pagination.
\ltrdoc	This document will have English-style pagination (the default).

Note that the three document-protection control words (**\formprot**, **\revprot**, and **\annotprot**) are mutually exclusive; only one of the three can apply to any given document. Also, there is currently no method for storing passwords in RTF, so any document that associates a password with a protection level will lose the password protection in RTF.

For more information about bidirectional controls, see “Bidirectional Language Support” on page 53 of this Application Note.

Section Text

Each section in the RTF file has the following syntax:

<section> <secfmt>* <hdrftr>? <para>+ (**sect** <section>)?

Section Formatting Properties

At the beginning of each section, there may be some section-formatting control words (described as <secfmt> in the section text syntax description). These control words specify section-formatting properties, which apply to the text *following* the control word, with the exception of the section-break control words (those beginning with **sbk**). Section-break control words describe the break *preceding* the text. These control words can appear anywhere in the section, not just at the start.

Note that if the **\sectd** control word is not present, the current section inherits all section properties defined in the previous section.

The section-formatting control words are listed in the following table.

Control word	Meaning
\sect	New section.
\sectd	Reset to default section properties.
\endnhere	Endnotes included in the section.
\binfsxn*N*	***N*** is the printer bin used for the first page of the section. If this control is not defined then the first page uses the same printer bin as defined by the **binsxn*N*** control.

\binsxn*N*	***N*** is the printer bin used for the pages of the section.
\ds*N*	Designates section style; if a section style is specified, style properties must be specified with the section.
\pnseclvl*N*	Used for multilevel lists. This property sets the default numbering style for each corresponding **\pnlvl*N*** control word (bullets and numbering property for paragraphs) within that section. This is a destination control word.
\sectunlocked	This section is unlocked for forms.

Section Break

\sbknone	No section break.
\sbkcol	Section break starts a new column.
\sbkpage	Section break starts a new page (the default).
\sbkeven	Section break starts at an even page.
\sbkodd	Section break starts at an odd page.

Columns

\cols*N*	Number of columns for "snaking" (the default is 1).
\colsx*N*	Space between columns in twips (the default is 720).
\colno*N*	Column number to be formatted; used to specify formatting for variable-width columns.
\colsr*N*	Space to right of column in twips; used to specify formatting for variable-width columns.
\colw*N*	Width of column in twips; used to override the default constant width setting for variable-width columns.
\linebetcol	Line between columns.

Line Numbering

\linemod*N*	Line-number modulus amount to increase each line number (the default is 1).
\linex*N*	Distance from the line number to the left text margin in twips (the default is 360). The automatic distance is 0.
\linestarts*N*	Beginning line number (the default is 1).
\linerestart	Line numbers restart at **linestarts** value.
\lineppage	Line numbers restart on each page.
\linecont	Line numbers continue from the preceding section.

Page Information

\pgwsxn*N*	***N*** is the page width in twips. A **\sectd** resets the value to that specified by **\paperw*N*** in the document properties.
\pghsxn*N*	***N*** is the page height in twips. A **\sectd** resets the value to that specified by **\paperh*N*** in the document properties.
\marglsxn*N*	***N*** is the left margin of the page in twips. A **\sectd** resets the value to that specified by **\margl*N*** in the document properties.
\margrsxn*N*	***N*** is the right margin of the page in twips. A **\sectd** resets the value to that specified by **\margr*N*** in the document properties.
\margtsxn*N*	***N*** is the top margin of the page in twips. A **\sectd** resets the value to that specified by **\margt*N*** in the document properties.

\margbsxn*N*	***N*** is the bottom margin of the page in twips. A **\sectd** resets the value to that specified by **\margb*N*** in the document properties.
\guttersxn*N*	***N*** is the width of the gutter margin for the section in twips. A **\sectd** resets the value to that specified by **\gutter*N*** from the document properties. If Facing Pages is turned off, the gutter will be added to the left margin of all pages. If Facing Pages is turned on, the gutter will be added to the left side of odd-numbered pages and the right side of even-numbered pages.
\lndscpsxn	Page orientation is in landscape format. To mix portrait and landscape sections within a document, the **\landscape** control should not be used so that the default for a section is portrait, which may be overridden by the **\lndscpsxn** control.
\titlepg	First page has a special format.
\header*y*N*	Header is ***N*** twips from the top of the page (the default is 720).
\footery*N*	Footer is ***N*** twips from the bottom of the page (the default is 720).

Page Numbers

\pgnstarts*N*	Beginning page number (the default is 1).
\pgncont	Continuous page numbering (the default).
\pgnrestart	Page numbers restart at **pgnstarts** value.
\pgnx*N*	Page number is ***N*** twips from the right margin (the default is 720).
\pgny*N*	Page number is ***N*** twips from the top margin (the default is 720).
\pgndec	Page-number format is decimal.
\pgnucrm	Page-number format is uppercase roman numeral.
\pgnlcrm	Page-number format is lowercase roman numeral.
\pgnucltr	Page-number format is uppercase letter.
\pgnlcltr	Page-number format is lowercase letter.
\pgnhn*N*	Indicates which heading level is used to prefix a heading number to the page number. This control word can only be used in conjunction with numbered heading styles. 0 specifies to not show heading level (the default). Values 1-9 correspond to heading levels 1 through 9.
\pgnhnsh	Hyphen separator character. This separator and the successive ones appear between the heading level number and the page number.
\pgnhnsp	Period separator character.
\pgnhnsc	Colon separator character.
\pgnhnsm	Em-dash (—) separator character.
\pgnhnsn	En-dash (–) separator character.

Vertical Alignment

\vertalt	Text is top-aligned (the default).
\vertalb	Text is bottom-aligned.
\vertalc	Text is centered vertically.
\vertalj	Text is justified vertically.

Bidirectional Controls

\rtlsect	This section will snake (newspaper style) columns from right to left.

\ltrsect	This section will snake (newspaper style) columns from left to right (the default).

Headers and Footers

Headers and footers are RTF destinations. Each section in the document can have its own set of headers and footers. If no headers or footers are defined for a given section, the headers and footers from the previous section (if any) are used. Headers and footers have the following syntax:

<hdrftr>	'{' <hdrctl> <para>+ '}' <hdrftr>?
<hdrctl>	**\header** | **\footer** | **\headerl** | **\headerr** | **\headerf** | **\footerl** | **\footerr** | **\footerf**

Note that each separate <hdrftr> group must have a distinct <hdrctl> introducing it.

Control word	Meaning
\header	Header on all pages. This is a destination control word.
\footer	Footer on all pages. This is a destination control word.
\headerl	Header on left pages only. This is a destination control word.
\headerr	Header on right pages only. This is a destination control word.
\headerf	Header on first page only. This is a destination control word.
\footerl	Footer on left pages only. This is a destination control word.
\footerr	Footer on right pages only. This is a destination control word.
\footerf	Footer on first page only. This is a destination control word.

The **headerl**, **headerr**, **footerl,** and **footerr** control words are used in conjunction with the **facingp** control word, and the **headerf** and **footerf** control words are used in conjunction with the **titlepg** control word. Many RTF readers will not function correctly if the appropriate document properties are not set. In particular, if **\facingp** is not set, then only **\header** and **\footer** should be used; if **\facingp** is set, then only **\headerl**, **\headerr**, **\footerl**, and **\footerr** should be used. Combining both **\facingp** and **\titlepg** is allowed. You should not use **\header** to set the headers for both pages when **\facingp** is set. You can use **\headerf** if **\titlepg** is not set, but no header will appear. For more information, see "{\info{\title The Panda's Thumb}{\author Stephen J Gould}{\keywords " on page 17 and "Section Formatting Properties" on page 20 of this Application Note.

If the previous section had a first page header or footer and had **titlepg** set, and the current section does not, then the previous section's first page header or footer is disabled. However, it is not destroyed; if subsequent sections have **titlepg** set, then the first page header or footer is restored.

Paragraph Text

There are two kinds of paragraphs: plain and table. A table is a collection of paragraphs, and a table row is a continuous sequence of paragraphs partitioned into cells. The **intbl** paragraph-formatting control word identifies the paragraph as part of a table. For more information, see "Table Definitions" on page 30 of this Application Note. This control is inherited between paragraphs that do not have paragraph properties reset with **pard**.

<para>	<textpar> | <row>
<textpar>	<pn>? <brdrdef>? <parfmt>* <apoctl>* <tabdef>? <shading>? (***subdocument** | <char>+) (**par** <para>)?
<row>	<tbldef> <cell>+ **\row**
<cell>	<textpar>+ **\cell**

Paragraph Formatting Properties

These control words (described as <parfmt> in the paragraph-text syntax description) specify generic paragraph formatting properties. These control words can appear anywhere in the body of the paragraph, not just at the beginning.

Note that if the **\pard** control word is not present, the current paragraph inherits all paragraph properties defined in the previous paragraph.

The paragraph-formatting control words are listed in the following table.

Control word	Meaning
\par	New paragraph.
\pard	Resets to default paragraph properties.
\s*N*	Designates paragraph style; if a paragraph style is specified, style properties must be specified with the paragraph. ***N*** references an entry in the stylesheet.
\hyphpar	Toggles automatic hyphenation for the paragraph. Append 1 or nothing to toggle property on; append 0 (zero) to turn it off.
\intbl	Paragraph is part of a table.
\keep	Keep paragraph intact.
\nowidctlpar	No widow/orphan control. This is a paragraph-level property and is used to override the document-level **\widowctrl**.
\keepn	Keep paragraph with the next paragraph.
\level*N*	***N*** is the outline level of the paragraph.
\noline	No line numbering.
\pagebb	Break page before the paragraph.
\sbys	Side-by-side paragraphs.
Alignment	
\ql	Left-aligned (the default).
\qr	Right-aligned.
\qj	Justified.
\qc	Centered.
Indentation	
\fi*N*	First-line indent (the default is 0).
\li*N*	Left indent (the default is 0).
\ri*N*	Right indent (the default is 0).
Spacing	
\sb*N*	Space before (the default is 0).
\sa*N*	Space after (the default is 0).
\sl*N*	Space between lines: if this control word is missing or if \ **sl1000** is used, the line spacing is automatically determined by the tallest character in the line; if ***N*** is a positive value, this size is used only if it is taller than the tallest character (otherwise the tallest character is used); if ***N*** is a negative value, the absolute value of ***N*** is used, even if it is shorter than the tallest character.

\slmult*N*	Line spacing multiple; indicates that the current line spacing is a multiple of "Single" line spacing. This control word can follow only the **sl** control word and works in conjunction with it. 0 "At Least" or "Exactly" line spacing. 1 Multiple line spacing, relative to "Single."

Subdocuments	
\subdocument*N*	This indicates that a subdocument in a master document/subdocument relationship should occur here. ***N*** represents an index into the file table. This control word must be the only item in a paragraph.

Bidirectional Controls	
\rtlpar	Text in this paragraph will be displayed with right-to-left precedence.
\ltrpar	Text in this paragraph will be displayed with left-to-right precedence (the default).

Tabs

Any paragraph may have its own set of tabs. Tabs must follow this syntax:

<tabdef>	(<tab> | <bartab>) +
<tab>	<tabkind>? <tablead>? ***tx***
<bartab>	<tablead>? ***tb***
<tabkind>	**\tqr** | **\tqc** | **\tqdec**
<tablead>	**\tldot** | **\tlhyph** | **\tlul** | **\tlth** | **\tleq**

Control word	Meaning
\tx*N*	Tab position in twips from the left margin.
\tqr	Flush-right tab.
\tqc	Centered tab.
\tqdec	Decimal tab.
\tb*N*	Bar tab position in twips from the left margin.
\tldot	Leader dots.
\tlhyph	Leader hyphens.
\tlul	Leader underline.
\tlth	Leader thick line.
\tleq	Leader equal sign.

Bullets and Numbering

To provide compatibility with existing RTF readers, all applications that can automatically bullet or number paragraphs will also emit the generated text as plain text in the **pntext** group. This will allow existing RTF readers to capture the plain text and safely ignore the autonumber instructions. This group precedes all bulleted or numbered paragraphs, and will contain all the text and formatting that would be auto-generated. It should precede the '**{*\pn ⋯ }**' destination, and it is the responsibility of RTF readers that understand the '**{*\pn ⋯ }**' destination to ignore the **pntext** group.

<pn>	<pnseclvl> | <pnpara>

<pnseclvl>	'{*' ***pnseclvl** <pndesc>'}'
<pnpara>	<pntext> <pnprops>
<pntext>	'{' **pntext** <char> '}'
<pnprops>	'{*' **pn** <pnlevel> <pndesc>'}'
<pnlevel>	***pnlvl*** | **pnlvlblt** | **pnlvlbody** | **pnlvlcont**
<pndesc>	<pnnstyle> & <pnchrfmt> & <pntxtb> & <pntxta> & <pnfmt>
<pnnstyle>	**pncard** | **pndec** | **pnucltr** | **pnucrm** | **pnlcltr** | **pnlcrm** | **pnord** | **pnordt**
<pnchrfmt>	***pnf***? & ***pnfs***? & **pnb**? & **pni**? & **pncaps**? & **pnscaps**? & <pnul>? & **pnstrike**? & ***pncf***?
<pnul>	**pnul** | **pnuld** | **pnuldb** | **pnulnone** | **pnulw**
<pnfmt>	**pnnumonce**? & **pnacross**? & ***pnindent***? & ***pnsp***? & **pnprev**? & <pnjust>? & ***pnstart***? & **pnhang**? & **pnrestart**?
<pnjust>	**pnqc** | **pnql** | **pnqr**
<pntxtb>	'{' **pntxtb** #PCDATA'}'
<pntxta>	'{' **pntxta** #PCDATA'}'

Settings below marked with an asterisk can be turned off by appending 0 (zero) to the control word.

Control word	Definition
\pntext	This group precedes all numbered/bulleted paragraphs, and contains all auto-generated text and formatting. It should precede the **'{*'\pn ⋯ '}'** destination, and it is the responsibility of RTF readers that understand the **'{*'\pn ⋯ '}'** destination to ignore this preceding group. This is a destination control word.
\pn	Turns on paragraph numbering. This is a destination control word.
\pnlvl*N*	Paragraph level, where ***N*** is a level from 1 to 9. Default set by **\pnseclvl*N*** section formatting property.
\pnlvlblt	Bulleted paragraph (corresponds to level 11). The actual character used for the bullet is stored in the **\pntxtb** group.
\pnlvlbody	Simple paragraph numbering (corresponds to level 10).
\pnlvlcont	Continue numbering but do not display number (“skip numbering”).
\pnnumonce	Number each cell only once in a table (the default is to number each paragraph in a table).
\pnacross	Number across rows (the default is to number down columns).
\pnhang	Paragraph uses a hanging indent.
\pnrestart	Restart numbering after each section break. Note that this control word is used only in conjunction with the Heading Numbering feature (applying multilevel numbering to Heading style definitions).
\pncard	Cardinal numbering (One, Two, Three).
\pndec	Decimal numbering (1, 2, 3).
\pnucltr	Uppercase alphabetic numbering (A, B, C).
\pnucrm	Uppercase roman numbering (I, II, III).
\pnlcltr	Lowercase alphabetic numbering (a, b, c).
\pnlcrm	Lowercase roman numbering. (i, ii, iii).

\pnord	Ordinal numbering (1st, 2nd, 3rd).
\pnordt	Ordinal text numbering (First, Second, Third).
\pnb	Bold numbering.*
\pni	Italic numbering.*
\pncaps	All Caps numbering.*
\pnscaps	Small Caps numbering.*
\pnul	Continuous underline.*
\pnuld	Dotted underline.
\pnuldb	Double underline.
\pnulnone	Turns off underlining.
\pnulw	Word underline.
\pnstrike	Strikethrough numbering.*
\pncf*N*	Foreground color—index into color table (the default is 0).
\pnf*N*	Font number.
\pnfs*N*	Font size (in half-points).
\pnindent*N*	Minimum distance from margin to body text.
\pnsp*N*	Distance from number text to body text.
\pnprev	Used for multilevel lists. Include information from previous level in this level; for example, 1, 1.1, 1.1.1, 1.1.1.1
\pnqc	Centered numbering.
\pnql	Left-justified numbering.
\pnqr	Right-justified numbering.
\pnstart*N*	Start At number.
\pntxta	Text after. This group contains the text that succeeds the number. This is a destination control word.
\pntxtb	Text before. This group contains the text that precedes the number. This is a destination control word.

Note that there is a limit of 32 characters total for the sum of text before and text after for simple numbering. Multilevel numbering has a limit of 64 characters total for the sum of all levels.

Paragraph Borders

Paragraph borders have the following syntax:

<brdrdef>	(<brdrseg> <brdr>)+
<brdrseg>	**\brdrt** | **\brdrb** | **\brdrl** | **\brdrr** | **\brdrbtw** | **\brdrbar** | **\box**
<brdr>	<brdrk> ***brdrw*** ? ***brsp*** ? ***brdrcf*** ?
<brdrk>	**\brdrs** | **\brdrth** | **\brdrsh** | **\brdrdb** | **\brdrdot** | **\brdrdash** | **\brdrhair**

Control word	**Meaning**

\brdrt	Border top.
\brdrb	Border bottom.
\brdrl	Border left.
\brdrr	Border right.
\brdrbtw	Consecutive paragraphs with identical border formatting are considered part of a single group with the border information applying to the entire group. To have borders around individual paragraphs within the group, the **\brdrbtw** control must be specified for that paragraph.
\brdrbar	Border outside (right side of odd-numbered pages, left side of even-numbered pages).
\box	Border around the paragraph (box paragraph).
\brdrs	Single-thickness border.
\brdrth	Double-thickness border.
\brdrsh	Shadowed border.
\brdrdb	Double border.
\brdrdot	Dotted border.
\brdrdash	Dashed border.
\brdrhair	Hairline border.
\brdrw*N*	***N*** is the width in twips of the pen used to draw the paragraph border line.
\brdrcf*N*	***N*** is the color of the paragraph border; specified as an index into the color table in the RTF header.
\brsp*N*	Space in twips between borders and the paragraph.

Paragraph Shading

Paragraph shading has the following syntax:

<shading>	(\ ***shading*** | <pat>) **\ *cfpat*** ? **\ *cbpat*** ?
<pat>	**\bghoriz** | **\bgvert** | **\bgfdiag** | **\bgbdiag** | **\bgcross** | **\bgdcross** | **\bgdkhoriz** | **\bgdkvert** | **\bgdkfdiag** | **\bgdkbdiag** | **\bgdkcross** | **\bgdkdcross**

Control word	Meaning
\shading*N*	***N*** is the shading of the paragraph in hundredths of a percent.
\bghoriz	Specifies a horizontal background pattern for the paragraph.
\bgvert	Specifies a vertical background pattern for the paragraph.
\bgfdiag	Specifies a forward diagonal background pattern for the paragraph (\\\\).
\bgbdiag	Specifies a backward diagonal background pattern for the paragraph (////).
\bgcross	Specifies a cross background pattern for the paragraph.
\bgdcross	Specifies a diagonal cross background pattern for the paragraph.
\bgdkhoriz	Specifies a dark horizontal background pattern for the paragraph.
\bgdkvert	Specifies a dark vertical background pattern for the paragraph.
\bgdkfdiag	Specifies a dark forward diagonal background pattern for the paragraph (****).

\bgdkbdiag	Specifies a dark backward diagonal background pattern for the paragraph (*////*).
\bgdkcross	Specifies a dark cross background pattern for the paragraph.
\bgdkdcross	Specifies a dark diagonal cross background pattern for the paragraph.
\cfpat*N*	***N*** is the line color of the background pattern, specified as an index into the document's color table.
\cbpat*N*	***N*** is the background color of the background pattern, specified as an index into the document's color table.

Positioned Objects and Frames

The following paragraph-formatting control words specify the location of a paragraph on the page. Consecutive paragraphs with the same frame formatting are considered to be part of the same frame. For two framed paragraphs to appear at the same position on a page, they must be separated by a paragraph with different or no frame information.

Note that if any paragraph in a table row has any of these control words specified, then all paragraphs in the table row must have the same control words specified, either by inheriting the properties from the previous paragraph or by respecifying the controls.

Paragraph positioning has the following syntax:

<apoctl>	<framesize> & <horzpos> & <vertpos> & <txtwrap> & <dropcap>
<framesize>	***absw***? & ***absh***?
<horzpos>	<hframe> & <hdist>
<vertpos>	<vframe> & <vdist>
<txtwrap>	**\nowrap**? & ***dxfrtext***? & ***dfrmtxtx***? &***dfrmtxty***?
<dropcap>	***dropcapli***? & ***dropcapt***?
<hframe>	**\phmrg**? | **\phpg**? | **\phcol**?
<hdist>	***posx***? | ***posnegx***? | **\posxc**? | **\posxi**? | **\posxo**? | **\posxl**? | **\posxr**?
<vframe>	**\pvmrg**? | **\pvpg**? | **\pvpara**?
<vdist>	***posy***? | ***posnegy***? | **\posyt**? | **\posyil**? | **\posyb**? | **\posyc**?

Control word	**Meaning**
\absw*N*	***N*** is the width of the frame in twips.
\absh*N*	***N*** is the height of the frame in twips. A positive number indicates the minimum height of the frame and a negative number indicates the exact height of the frame. A value of zero indicates that the height of the frame adjusts to the contents of the frame. This is the default for frames where no height is given.
Horizontal Position	
\phmrg	Use the margin as the horizontal reference frame.
\phpg	Use the page as the horizontal reference frame.
\phcol	Use the column as the horizontal reference frame. This is the default if no horizontal reference frame is given.
\posx*N*	Positions the frame ***N*** twips from the left edge of the reference frame.
\posnegx*N*	Same as **\posx** but allows arbitrary negative values.

\posxc	Centers the frame horizontally within the reference frame.
\posxi	Positions the paragraph horizontally inside the reference frame.
\posxo	Positions the paragraph horizontally outside the reference frame.
\posxr	Positions the paragraph to the right within the reference frame.
\posxl	Positions the paragraph to the left within the reference frame. This is the default if no horizontal positioning information is given.

Vertical Position

\pvmrg	Positions the reference frame vertically relative to the margin. This is the default if no vertical frame positioning information is given.
\pvpg	Positions the reference frame vertically relative to the page.
\pvpara	Positions the reference frame vertically relative to the top of the top left corner of the next unframed paragraph in the RTF stream.
\posy*N*	Positions the paragraph ***N*** twips from the top edge of the reference frame.
\posnegy*N*	Same as **\posy** but allows arbitrary negative values.
\posyil	Positions the paragraph vertically to be in-line.
\posyt	Positions the paragraph at the top of the reference frame.
\posyc	Centers the paragraph vertically within the reference frame.
\posyb	Positions the paragraph at the bottom of the reference frame.

Text Wrapping

\nowrap	Prevents text from flowing around the positioned object.
\dxfrtext*N*	Distance in twips of a positioned paragraph from text in the main text flow in all directions.
\dfrmtxtx*N*	***N*** is the horizontal distance in twips from text on both sides of the frame.
\dfrmtxty*N*	***N*** is the vertical distance in twips from text on both sides of the frame.

Drop Caps

\dropcapli*N*	Number of lines drop cap is to occupy. The range is 1 through 10.
\dropcapt*N*	Type of drop cap: 1 In-text drop cap 2 Margin drop cap

The following is an example of absolute-positioned text in a document:

```
\par \pard \pvpg\phpg\posxc\posyt\absw5040\dxfrtest173 First APO para
\par \pard \phmrg\posxo\posyc\dxfrtext1152 Second APO para
```

Table Definitions

There is no RTF table group; instead, tables are specified as paragraph properties. A table is represented as a sequence of table rows. A table row is a continuous sequence of paragraphs partitioned into cells. The table row begins with the **trowd** control word and ends with the **row** control word. Every paragraph that is contained in a table row must have the **intbl** control word specified or inherited from the previous paragraph. A cell may have more than one paragraph in it; the cell is terminated by a cell mark (the **cell** control word), and the row is terminated by a row mark (the **row** control word). Table rows can also be positioned. In this case, every paragraph in a table row must have the same positioning controls (see the <apoctl> controls on page 29 of this

Application Note). Table properties may be inherited from the previous row; therefore, a series of table rows may be introduced by a single <tbldef>.

An RTF table row has the following syntax, as shown in the general paragraph-text syntax shown on page 23 of this Application Note.

<row>	<tbldef> <cell>+ **\row**
<cell>	<textpar>+ **\cell**

A table definition has the following syntax:

<tbldef>	**\trowd *trgaph*** <rowjust>? & <rowwrite>? <rowtop>? & <rowbot>? & <rowleft>? & <rowright>? & <rowhor>? & <rowvert>? & ***trleft***? & ***trrh***? **\trhdr**? & **\trkeep**? <celldef>+
<rowjust>	**\trql** | **\trqr** | **\trqc**
<rowwrite>	**\ltrrow** | **\rtlrow**
<rowtop>	**\trbrdrt** <brdr>
<rowbot>	**\trbrdrl** <brdr>
<rowleft>	**\trbrdrb** <brdr>
<rowright>	**\trbrdrr** <brdr>
<rowhor>	**\trbrdrh** <brdr>
<rowvert>	**\trbrdrv** <brdr>
<celldef>	(**clmgf**? & **\clmrg**? <celltop>? & <cellleft>? & <cellbot>? & <cellright>? & <cellshad>?) ***cellx***
<celltop>	**\clbrdrt** <brdr>
<cellleft>	**\clbrdrl** <brdr>
<cellbot>	**\clbrdrb** <brdr>
<cellright>	**\clbrdrr** <brdr>
<cellshad>	<cellpat>? ***clcfpat***? & ***clcbpat***? & ***clshdng***
<cellpat>	**\clbghoriz** | **\clbgvert** | **\clbgfdiag** | **\clbgbdiag** | **\clbgcross** | **\clbgdcross** | **\clbgdkhor** | **\clbgdkvert** | **\clbgdkfdiag** | **\clbgdkbdiag** | **\clbgdkcross** | **\clbgdkdcross**

Note for <tbldef> that the number of **\cellx**s must match the number of **cell**s in the **row**.

The following control words further define options for each row of the table.

Control word	**Meaning**
\trowd	Sets table row defaults.
\trgaph*N*	Half the space between the cells of a table row in twips.
\cellx*N*	Defines the right boundary of a table cell, including its half of the space between cells.
\clmgf	The first cell in a range of table cells to be merged.
\clmrg	Contents of the table cell are merged with those of the preceding cell.
Row Formatting	
\trql	Left-justifies a table row with respect to its containing column.
\trqr	Right-justifies a table row with respect to its containing column.
\trqc	Centers a table row with respect to its containing column.

\trleft*N* Position of the leftmost edge of the table with respect to the left edge of its column.

\trrh*N* Height of a table row in twips; when 0 (zero), the height is sufficient for all the text in the line; when positive, the height is guaranteed to be at least the specified height; when negative, the absolute value of the height is used, regardless of the height of the text in the line.

\trhdr Table row header; this row should appear at the top of every page the current table appears on.

\trkeep Table row keep together; this row cannot be split by a page break. This property is assumed off unless the control word is present.

Bidirectional Controls

\rtlrow Cells in this table row will have right-to-left precedence.

\ltrrow Cells in this table row will have left-to-right precedence (the default).

Row Borders

\trbrdrt Table row border top.

\trbrdrl Table row border left.

\trbrdrb Table row border bottom.

\trbrdrr Table row border right.

\trbrdrh Table row border horizontal (inside).

\trbrdrv Table row border vertical (inside).

Cell Borders

\clbrdrb Bottom table cell border.

\clbrdrt Top table cell border.

\clbrdrl Left table cell border.

\clbrdrr Right table cell border.

Cell Shading and Background Pattern

\clshdng*N* ***N*** is the shading of a table cell in hundredths of a percent. This control should be included in RTF along with cell border information.

\clbghoriz Specifies a horizontal background pattern for the cell.

\clbgvert Specifies a vertical background pattern for the cell.

\clbgfdiag Specifies a forward diagonal background pattern for the cell (\\\\).

\clbgbdiag Specifies a backward diagonal background pattern for the cell (////).

\clbgcross Specifies a cross background pattern for the cell.

\clbgdcross Specifies a diagonal cross background pattern for the cell.

\clbgdkhor Specifies a dark horizontal background pattern for the cell.

\clbgdkvert Specifies a dark vertical background pattern for the cell.

\clbgdkfdiag Specifies a dark forward diagonal background pattern for the cell (****).

\clbgdkbdiag Specifies a dark backward diagonal background pattern for the cell (***////***).

\clbgdkcross Specifies a dark cross background pattern for the cell.

\clbgdkdcross	Specifies a dark diagonal cross background pattern for the cell.
\clcfpat*N*	***N*** is the line color of the background pattern.
\clcbpat*N*	***N*** is the background color of the background pattern.

The following is an example of table text:

```
\par \trowd \trqc\trgaph108\trrh280\trleft36
\clbrdrt\brdrth \clbrdrl\brdrth \clbrdrb\brdrdb
\clbrdrr\brdrdb \cellx3636\clbrdrt\brdrth
\clbrdrl\brdrdb \clbrdrb\brdrdb \clbrdrr\brdrdb
\cellx7236\clbrdrt\brdrth \clbrdrl\brdrdb
\clbrdrb\brdrdb \clbrdrr\brdrdb \cellx10836\pard \intbl
\cell \pard \intbl \cell \pard \intbl \cell \pard \intbl \row
\trowd \trqc\trgaph108\trrh280\trleft36 \clbrdrt\brdrdb
\clbrdrl\brdrth \clbrdrb \brdrsh\brdrs \clbrdrr\brdrdb
\cellx3636\clbrdrt\brdrdb \clbrdr \brdrdb
\clbrdrb\brdrsh\brdrs \clbrdrr\brdrdb
\cellx7236\clbrdrt\brdrdb \clbrdr \brdrdb
\clbrdrb\brdrsh\brdrs \clbrdrr\brdrdb \cellx10836\pard
\intbl \cell \pard \intbl \cell \pard \intbl \cell \pard
\intbl \row \pard
```

Character Text

Character text has the following syntax:

<char>	<ptext> | <atext> | '{' <char> '}'
<ptext>	(<chrfmt>* <data>+)+
<data>	#PCDATA | <spec> | <pict> | <obj> | <do> | <foot> | <annot> | <field> | <idx> | <toc> | <book>

Character Formatting Properties

These control words (described as <chrfmt> in the syntax description) change character formatting properties. A control word preceding plain text turns on the specified attribute. Some control words (indicated in the following table by an asterisk following the description) can be turned off by the control word followed by 0 (zero). For example**, \b** turns on bold, while **b0** turns off bold.

The character-formatting control words are listed in the following table.

Control word	Meaning
\plain	Reset character formatting properties to a default value defined by the application. The associated character formatting properties (described in the section "Associated Character Properties" on page 36 of this Application Note) are also reset.
\b	Bold.*
\caps	All capitals.*
\deleted	Marks the text as deletion revision marked.*
\dn*N*	Subscript position in half-points (the default is 6).
\sub	Subscripts text and shrinks point size according to font information.
\nosupersub	Turns off superscripting or subscripting.
\expnd*N*	Expansion or compression of the space between characters in quarter-points; a negative value compresses (the default is 0).

\expndtw*N*	Expansion or compression of the space between characters in twips; a negative value compresses. For backward compatibility, both **\expndtw** and **\expnd** should be emitted.
\kerning*N*	Point size (in half-points) above which to kern character pairs. **\kerning0** turns off kerning.
\f*N*	Font number. ***N*** refers to an entry in the font table.
\fs*N*	Font size in half-points (the default is 24).
\i	Italic.*
\outl	Outline.*
\revised	Text has been added since revision marking was turned on.
\revauth*N*	Index into the revision table. The content of the ***N***th group in the revision table is considered to be the author of that revision.
\revdttm*N*	Time of the revision. The 32-bit DTTM structure is emitted as a long integer.
\scaps	Small capitals.*
\shad	Shadow.*
\strike	Strikethrough.*
\ul	Continuous underline. **\ul0** turns off all underlining.
\uld	Dotted underline.
\uldb	Double underline.
\ulnone	Stops all underlining.
\ulw	Word underline.
\up*N*	Superscript position in half-points (the default is 6).
\super	Superscripts text and shrinks point size according to font information.
\v	Hidden text.*
\cf*N*	Foreground color (the default is 0).
\cb*N*	Background color (the default is 0).
\rtlch	The character data following this control word will be treated as a right-to-left run.
\ltrch	The character data following this control word will be treated as a left-to-right run (the default).
\cs*N*	Designates character style; if a character style is specified, style properties must be specified with the character run. ***N*** refers to an entry in the style table.
\cchs*N*	Indicates any characters not belonging to the default document character set and tells which character set they do belong to. Macintosh character sets are represented by values greater than 255. The values for ***N*** correspond to the values for the **\fcharset** control word.
\lang*N*	Applies a language to a character. ***N*** is a number corresponding to a language. The **plain** control word resets the language property to the language defined by **deflang*N*** in the document properties.

The following table defines the standard languages used by Microsoft. This table was generated by the Unicode™ group for use with TrueType and Unicode.

Language name	Language ID
No language	0x0400

Albanian	0x041c
Arabic	0x0401
Bahasa	0x0421
Belgian Dutch	0x0813
Belgian French	0x080c
Brazilian Portuguese	0x0416
Bulgarian	0x0402
Catalan	0x0403
Croato-Serbian (Latin)	0x041a
Czech	0x0405
Danish	0x0406
Dutch	0x0413
English (Australian)	0x0c09
English (U.K.)	0x0809
English (U.S.)	0x0409
Finnish	0x040b
French	0x040c
French (Canadian)	0x0c0c
German	0x0407
Greek	0x0408
Hebrew	0x040d
Hungarian	0x040e
Icelandic	0x040f
Italian	0x0410
Japanese	0x0411
Korean	0x0412
Norwegian (Bokmal)	0x0414
Norwegian (Nynorsk)	0x0814
Polish	0x0415
Portuguese	0x0816
Rhaeto-Romanic	0x0417
Romanian	0x0418
Russian	0x0419
Serbo-Croatian (Cyrillic)	0x081a
Simplified Chinese	0x0804
Slovak	0x041b
Spanish (Castilian)	0x040a
Spanish (Mexican)	0x080a

Swedish	0x041d
Swiss French	0x100c
Swiss German	0x0807
Swiss Italian	0x0810
Thai	0x041e
Traditional Chinese	0x0404
Turkish	0x041f
Urdu	0x0420

To read negative **expnd** values from Word for the Macintosh, an RTF reader should use only the low-order 6 bits of the value read. Word for the Macintosh does not emit negative values for **expnd**. Instead, it treats values from 57 through 63 as –7 through –1, respectively (the low-order 6 bits of 57 through 63 are the same as –7 through –1).

Associated Character Properties

Bidirectional-aware text processors often need to associate a Latin (or other left-to-right) font with an Arabic or Hebrew (or other right-to-left) font. The association is needed to match commonly used pairs of fonts in name, size, and other attributes. While RTF defines a broad variety of associated character properties, any implementation may choose to not implement a particular associated character property and share the property between the Latin and Arabic fonts.

Property association uses the following syntax:

<atext>	<ltrrun> | <rtlrun>
<ltrrun>	**\rtlch *af** & <aprops>* **\ltrch** <ptext>
<rtlrun>	**\ltrch *af** & <aprops>* **\rtlch** <ptext>

Here are some examples of property association:

```
\ ltrch\ af2\ ab\ au\ rtlch\ u Sample Text
```

This is a right-to-left run. Text will use the default bidirectional font, and will be underlined. The left-to-right font associated with this run is font 2 (in the font table) with bolding and underlining.

```
\ plain\ rtlch\ ltrch Sample Text
```

This is a left-to-right run. The right-to-left font and the left-to-right font use the default font (specified by **deff**).

```
\ rtlch\ af5\ ab\ ai\ ltrch\ u Sample Text
```

This is a left-to-right run. The right-to-left font is font 5, bold and italicized. The left-to-right font is the default font, underlined. If the reader does not support underlining in the associated font, both fonts will be underlined.

The property association control words (described as <aprops> in the syntax description) are listed in the following table. Some control words (indicated in the following table by an asterisk following the description) can be turned off by the control word followed by 0 (zero).

Control word	Meaning
\ab	Associated font is bold.*
\acaps	Associated font is all capitals.*
\acf*N*	Associated foreground color (the default is 0).
\adn*N*	Associated font is subscript position in half-points (the default is 6).

\aexpnd*N*	Expansion or compression of the space between characters in quarter-points; a negative value compresses (the default is 0).
\af*N*	Associated font number (the default is 0).
\afs*N*	Associated font size in half-points (the default is 24).
\ai	Associated font is italic.*
\alang*N*	Language ID for the associated font. (This uses the same language ID codes describedon page 34 of this Application Note.)
\aoutl	Associated font is outline.*
\ascaps	Associated font is small capitals.*
\ashad	Associated font is shadow.*
\astrike	Associated font is strikethrough.*
\aul	Associated font is continuous underline. **\aul0** turns off all underlining for the alternate font.
\auld	Associated font is dotted underline.
\auldb	Associated font is double underline.
\aulnone	Associated font is no longer underlined.
\aulw	Associated font is word underline.
\aup*N*	Superscript position in half-points (the default is 6).

Special Characters

The RTF Specification includes control words for special characters (described as <spec> in the character-text syntax description). If a special-character control word is not recognized by the RTF reader, it is ignored, and the text following it is considered plain text. The RTF Specification is flexible enough to allow new special characters to be added for interchange with other software.

The special RTF characters are listed in the following table.

Control word	Meaning
\chdate	Current date (as in headers).
\chdpl	Current date in long format (for example, Thursday, October 28, 1993).
\chdpa	Current date in abbreviated format (for example, Thu, Oct 28, 1993).
\chtime	Current time (as in headers).
\chpgn	Current page number (as in headers).
\sectnum	Current section number (as in headers).
\chftn	Automatic footnote reference (footnotes follow in a group).
\chatn	Annotation reference (annotation text follows in a group).
\chftnsep	Anchoring character for footnote separator.
\chftnsepc	Anchoring character for footnote continuation.
\cell	End of table cell.
\row	End of table row.
\par	End of paragraph.
\sect	End of section and paragraph.

\page	Required page break.
\column	Required column break.
\line	Required line break (no paragraph break).
\softpage	Nonrequired page break. Emitted as it appears in galley view.
\softcol	Nonrequired column break. Emitted as it appears in galley view.
\softline	Nonrequired line break. Emitted as it appears in galley view.
\softlheight*N*	Nonrequired line height. This is emitted as a prefix to each line.
\tab	Tab character; same as ASCII 9.
\emdash	Em-dash (—).
\endash	En-dash (–).
\emspace	Nonbreaking space equal to width of character "m" in current font.
\enspace	Nonbreaking space equal to width of character "n" in current font.
\bullet	Bullet character.
\lquote	Left single quotation mark.
\rquote	Right single quotation mark.
\ldblquote	Left double quotation mark.
\rdblquote	Right double quotation mark.
|	Formula character.
\~	Nonbreaking space.
\-	Optional hyphen.
_	Nonbreaking hyphen.
\:	Specifies a subentry in an index entry.
*****	Marks a destination whose text should be ignored if not understood by the RTF reader.
\'*hh*	A hexadecimal value, based on the specified character set (may be used to identify 8-bit values).
\ltrmark	The following characters should be displayed from left to right; usually found at the start of **\ltrch** runs.
\rtlmark	The following characters should be displayed from right to left; usually found at the start of **\rtlch** runs.
\zwj	Zero-width joiner. This is used to ligate (join) characters.
\zwnj	Zero-width nonjoiner. This is used for unligating a characters.

Note that an ASCII 9 is accepted as a tab character. A carriage return (character value 13) or linefeed (character value 10) will be treated as a **\par** control if the character is preceded by a backslash. You must include the backslash, or RTF ignores the control word. (You may also want to insert a carriage-return/linefeed pair without backslashes at least every 255 characters for better text transmission over communication lines.)

The following are the code values for the special characters listed.

Control word	Word for Windows and OS/2	Apple Macintosh
\bullet	149	0xA5
\endash	150	0xD1

\emdash	151	0xD0
\lquote	145	0xD4
\rquote	146	0xD5
\ldblquote	147	0xD2
\rdblquote	148	0xD3

Bookmarks

This destination may specify one of two control words: ***\bkmkstart**, which indicates the start of the specified bookmark, and ***\bkmkend**, which indicates the end of the specified bookmark.

Bookmarks have the following syntax:

<book>	<bookstart> | <bookend>
<bookstart>	'{*' **\bkmkstart** (***bkmkcolf**? & ***bkmkcoll**?) #PCDATA '}'
<bookend>	'{*' **\bkmkend** #PCDATA '}'

A bookmark is shown in the following example:

```
\pard\plain \fs20 Kuhn believes that science, rather than
discovering in experience certain structured
relationships, actually creates (or already participates in)
a presupposed structure to which it fits the data.
{\bkmkstart paradigm} Kuhn calls such a presupposed
structure a paradigm.{\bkmkend paradigm}
```

The bookmark start and the bookmark end are matched with the bookmark tag. In the example, the bookmark tag was "paradigm." Each bookmark start should have a matching bookmark end; however, the bookmark start and the bookmark end may be in any order.

\bkmkcolf*N* is used to denote the first column of a table covered by a bookmark. If it is not included, the first column is assumed. **bkmkcoll*N** is used to denote the last column. If it is not used, the last column is assumed. These controls are used within the ***\bkmkstart** destination following the **\bkmkstart** control. For example, `{\ *\ bkmkstart\ bkmkcolf2\ bkmkcoll5 Table1}` places the bookmark "Table1" on columns 2 through 5 of a table.

Pictures

An RTF file can include pictures created with other applications. These pictures can be in hexadecimal (the default) or binary format. Pictures are destinations, and begin with the **pict** control word. A picture destination has the following syntax:

<pict>	'{' **\pict** (<brdr>? & <shading>? & <picttype> & <pictsize> & <metafileinfo>?) <data> '}'
<picttype>	**\macpict** | ***pmmetafile*** | ***wmetafile*** | ***dibitmap*** <bitmapinfo> | ***wbitmap*** <bitmapinfo>
<bitmapinfo>	***wbmbitspixel*** & ***wbmplanes*** & ***wbmwidthbytes***
<pictsize>	(***picw*** & ***pich***) *picwgoal*? & *pichgoal*? **\picscalex**? & **\picscaley**? & **\picscaled**? & **\piccropt**? & **\piccropb**? & **\piccropr**? & **\piccropl**?
<metafileinfo>	**\picbmp** & ***picbpp***
<data>	(***bin*** #BDATA) | #SDATA

These control words are described in the following table (some measurements in this table are in twips; a twip is one-twentieth of a point).

Control word	Meaning
\macpict	Source of the picture is QuickDraw.
\pmmetafile*N*	Source of the picture is an OS/2 metafile; the ***N*** argument identifies the metafile type. The ***N*** values are descibed on page Error: Reference source not found of this Application Note.
\wmetafile*N*	Source of the picture is a Windows metafile; the ***N*** argument identifies the metafile type (the default is 1).
\dibitmap*N*	Source of the picture is a Windows device-independent bitmap; the ***N*** argument identifies the bitmap type (must equal 0). The information to be included in RTF from a Windows device-independent bitmap is the concatenation of the BITMAPINFO structure followed by the actual pixel data.
\wbitmap*N*	Source of the picture is a Windows device-dependent bitmap; the ***N*** argument identifies the bitmap type (must equal 0). The information to be included in RTF from a Windows device-dependent bitmap is the result of the GetBitmapBits function.

For more information on the GetDIBits and GetBitmapBits functions and the structure of Windows device-independent and device-dependent bitmaps, see *Volume 1* and *Volume 2* of the *Programmer's Reference* in the Microsoft Windows 3.1 Software Development Kit. For best device-independence and interoperability with Microsoft products, however, use of the **wbitmap** and **dibitmap** control words is discouraged. Rather, bitmaps should be embedded within Windows metafiles and the **wmetafile** control word used. For more information on embedding bitmaps within metafiles, see *Volume 1* and *Volume 2* of the *Programmer's Reference* in the Microsoft Windows 3.1 Software Development Kit.

Control word	Meaning
Bitmap Information	
\wbmbitspixel*N*	Number of adjacent color bits on each plane needed to define a pixel (the default is 1). Possible values are 1 (monochrome), 4 (16 colors), 8 (256 colors) and 24 (RGB).
\wbmplanes*N*	Number of bitmap color planes (must equal 1).
\wbmwidthbytes*N*	Specifies the number of bytes in each raster line. This value must be an even number since the Windows graphics device interface (GDI) assumes that the bit values of a bitmap form an array of integer (two-byte) values. In other words, **wbmwidthbytes** times 8 must be the next multiple of 16 greater than or equal to the **picw** (bitmap width in pixels) value.
Picture Size, Scaling, and Cropping	
\picw*N*	*xExt* field if the picture is a Windows metafile; picture width in pixels if the picture is a bitmap or from QuickDraw.
\pich*N*	*yExt* field if the picture is a Windows metafile; picture height in pixels if the picture is a bitmap or from QuickDraw.
\picwgoal*N*	Desired width of the picture in twips.
\pichgoal*N*	Desired height of the picture in twips.
\picscalex*N*	Horizontal scaling value; the ***N*** argument is a value representing a percentage (the default is 100).
\picscaley*N*	Vertical scaling value; the ***N*** argument is a value representing a percentage (the default is 100).
\picscaled	Scales the picture to fit within the specified frame; used only with **macpict** pictures.

\piccropt*N*	Top cropping value in twips; a positive value crops toward the center of the picture; a negative value crops away from the center, adding a space border around picture (the default is 0).
\piccropb*N*	Bottom cropping value in twips; a positive value crops toward the center of the picture; a negative value crops away from the center, adding a space border around picture (the default is 0).
\piccropl*N*	Left cropping value in twips; a positive value crops toward the center of the picture; a negative value crops away from the center, adding a space border around picture (the default is 0).
\piccropr*N*	Right cropping value in twips; a positive value crops toward the center of the picture; a negative value crops away from the center, adding a space border around picture (the default is 0).

Metafile Information

\picbmp	Specifies whether a metafile contains a bitmap.
\picbpp*N*	Specifies the bits per pixel in a metafile bitmap. The valid range is 1–32, with 1, 4, 8, and 24 being recognized.

Picture Data

\bin*N*	The picture is in binary format; the numeric parameter ***N*** is the number of bytes that follow. Unlike all other controls, this control word takes a 32-bit parameter.

The **wbitmap** control word is optional; if no other picture type is specified, the picture is assumed to be a Windows bitmap. If **wmetafile** is specified, the ***N*** argument can be one of the following types.

Type	***N* argument**
MM_TEXT	1
MM_LOMETRIC	2
MM_HIMETRIC	3
MM_LOENGLISH	4
MM_HIENGLISH	5
MM_TWIPS	6
MM_ISOTROPIC	7
MM_ANISOTROPIC	8

For more information about these types, see volume 1 of the *Programmer's Reference* in the Microsoft Windows 3.1 Software Development Kit.

If **pmmetafile** is specified, the ***N*** argument can be one of the following types.

Type	***N* argument**
PU_ARBITRARY	0x0004
PU_PELS	0x0008
PU_LOMETRIC	0x000C
PU_HIMETRIC	0x0010
PU_LOENGLISH	0x0014
PU_HIENGLISH	0x0018

PU_TWIPS	0x001C

For more information about these types, see volume 2 of the *OS/2 Programmer's Reference*.

Be careful with spaces following control words when dealing with pictures in binary format. When reading files, RTF considers the first space after a control word the delimiter and subsequent spaces part of the document text. Therefore, any extra spaces are attached to the picture, with unpredictable results.

RTF writers should not use the carriage-return/linefeed (CR/LF) combination to break up pictures in binary format. If they do, the CR/LF combination is treated as literal text and considered part of the picture data.

The picture in hexadecimal or binary format follows the picture-destination control words. The following example illustrates the destination format:

```
{\pict\wbitmap0\picw170\pich77\wbmbitspixel1\wbmplanes1\wbmwidthbytes22
\picwgoal505
\pichgoal221
\picscalex172
\picscaley172
49f2000000000273023d1101a030
3901000a000000000273023d98
0048000200000275
0204000020010275023e000000000
273023d000002b90002b90002
b90002b90002b9
0002b90002b90002b90002b90002b90002
b92222b90002b90002b90
002b90002b9
0002b90002b90002b90002b9000
```

Objects

Microsoft OLE links, Microsoft OLE embedded objects, and Macintosh Edition Manager subscriber objects are represented in RTF as objects. Objects are destinations that contain a data part and a result part. The data part is generally hidden to the application that produced the document. A separate application uses the data and supplies the appearance of the data. This appearance is the result part of the object.

The representation of objects in RTF is designed to allow RTF readers that don't understand objects or don't use a particular type of object to use the current result in place of the object. This allows the appearance of the object to be maintained through the conversion even though the object functionality is lost. Each object comes with optional information about the object, a required destination that contains the object data, and an optional result that contains the current appearance of the object. This result contains standard RTF. It is an important responsibility of the RTF writer to provide the result so that existing RTF readers that either do not support objects or that do not support the particular type of object will be able to display the object.

When the object is an OLE embedded or linked object, the data part of the object is the structure produced by the OLESaveToStream function. Some OLE clients rely on the OLE system to render the object and a copy of the result is not available to the RTF writer for that application. For these cases, the object result can be extracted from the structure produced by the OLESaveToStream function. For information about the OLESaveToStream function, see the Microsoft Object Linking and Embedding Software Development Kit.

The syntax for this destination is:

<obj>	('{' **\object** (<objtype> & <objmod>? & <objclass>? & <objname>? & <objtime>? & <objsize>? & <rsltmod>?) <objdata> <result> '}') | <pubobject>
<objtype>	**\objemb** | **\objlink** | **\objautlink** | **\objsub** | **\objpub** | **\objicemb**
<objmod>	**\linkself**? & **\objlock**? | **\objupdate**?
<objclass>	'{*' **\objclass** #PCDATA '}'
<objname>	'{*' **\objname** #PCDATA '}'
<objtime>	'{*' **\objtime** <time> '}'

<rsltmod>	**\rsltmerge**? & <rslttype>?
<rslttype>	**\rsltrtf** | **\rslttxt** | **\rsltpict** | **\rsltbmp**
<objsize>	**\objsetsize**? & ***objalign**? & ***objtransy**? & <objhw>? & ***objcropt**? & ***objcropb**? & ***objcropl**? & ***objcropr**? & ***objscalex**? & ***objscaley**?
<objhw>	***objh** & ***objw**
<objdata>	'{*' **\objdata** (<objalias>? & <objsect>?) <data> '}'
<objalias>	'{*' **\objalias** <data> '}'
<objsect>	'{*' **\objsect** <data> '}'
<result>	'{' **\result** <para>+ '}'

Control word	Meaning
Object Type	
\objemb	An object type of OLE embedded object. If no type is given for the object, the object is assumed to be of type **\objemb**.
\objlink	An object type of OLE link.
\objautlink	An object type of OLE autolink.
\objsub	An object type of Macintosh Edition Manager subscriber.
\objpub	An object type of Macintosh Edition Manager publisher.
\objicemb	An object type of MS® Word for the Macintosh Installable Command (IC) Embedder.
Object Information	
\linkself	The object is a link to another part of the same document.
\objlock	Locks the object from any updates.
\objupdate	Forces an update to the object before displaying it. Note that this will override any values in the <objsize> control words, but reasonable values should always be provided for these to maintain backwards compatibility.
\objclass	The text argument is the object class to use for this object; ignore the class specified in the object data. This is a destination control word.
\objname	The text argument is the name of this object. This is a destination control word.
\objtime	Describes the time that the object was last updated.
Object Size, Position, Cropping, and Scaling	
\objh*N*	***N*** is the original object height in twips, assuming the object has a graphical representation.
\objw*N*	***N*** is the original object width in twips, assuming the object has a graphical representation.
\objsetsize	Forces the object server to set the object's dimensions to that specified by the client.
\objalign*N*	***N*** is the distance in twips from the left edge of the objects that should be aligned on a tab stop. This is needed to place Equation Editor equations correctly in line.
\objtransy*N*	***N*** is the distance in twips the objects should be moved vertically with respect to the baseline. This is needed to place Math Type equations correctly in line.
\objcropt*N*	***N*** is the top cropping distance in twips.
\objcropb*N*	***N*** is the bottom cropping distance in twips.

\objcropl*N*	***N*** is the left cropping distance in twips.
\objcropr*N*	***N*** is the right cropping distance in twips.
\objscalex*N*	***N*** is the horizontal scaling percentage.
\objscaley*N*	***N*** is the vertical scaling percentage.

Object Data	
\objdata	This subdestination contains the data for the object in the appropriate format; OLE objects are in OLESaveToStream format. This is a destination control word.
\objalias	This subdestination contains the Alias Record for the publisher object for the Macintosh Edition Manager. This is a destination control word.
\objsect	This subdestination contains the Section Record for the publisher object for the Macintosh Edition Manager. This is a destination control word.

Object Result	
\rsltrtf	Forces the result to be rich-text format, if possible.
\rsltpict	Forces the result to be a Windows metafile or MacPict image format, if possible.
\rsltbmp	Forces the result to be a bitmap, if possible.
\rslttxt	Forces the result to be plain text, if possible.
\rsltmerge	Uses the formatting of the current result whenever a new result is obtained.
\result	The result destination is optional in the **\object** destination. It contains the last update of the result of the object. The data of the result destination should be standard RTF so that RTF readers that don't understand objects or the type of object represented can use the current result in the object's place to maintain appearance. This is a destination control word.

Macintosh Edition Manager Publisher Objects

Word for the Macintosh writes publisher objects for the Macintosh Edition Manager in terms of bookmarks (see "Bookmarks" on page 39 of this Application Note). The range of publisher objects are marked as bookmarks, so these controls are all used within the **\bkmkstart** destination. The RTF syntax for a publisher object is:

<pubobject> '{*' **\bkmkstart \bkmkpub \pubauto**? (<objalias>? & <objsect>) #PCDATA '}'

Control word	**Meaning**
\bkmkpub	The bookmark marks a Macintosh Edition Manager publisher object.
\pubauto	The publisher object updates all Macintosh Edition Manager subscribers of this object automatically whenever it is edited.

Drawing Objects

Drawing objects and the drawing primitives enumerated within drawing object groups use the syntax described by the following tables.

<do> '{*' **do** <dohead> <dpinfo>'}'

<dohead> <dobx> <doby> <dodhgt> <dolock>?

<dobx> **dobxpage** | **dobxcolumn** | **dobxmargin**

<doby> **dobypage** | **dobypara** | **dobymargin**

<dodhgt>	***dodhgt***
<dolock>	**dolock**
<dpinfo>	<dpgroup> | <dpcallout> | <dpsimple>
<dpgroup>	**\dpgroup *dpcount*** <dphead> <dpinfo>+ **\dpendgroup** <dphead>
<dpcallout>	**dpcallout** <cotype> <coangle>? <coaccent>? <cosmartattach>? <cobestfit>? <cominusx>? <cominusy>? <coborder>? <codescent>? ***dpcooffset*** ***dpcolength*** <dphead> <dppolyline> <dphead> <dpprops> <dptextbox> <dphead> <dpprops>
<dpsimple>	<dpsimpledpk> <dphead> <dpprops>
<dpsimpledpk>	<dpline> | <dprect> | <dptextbox> | <dpellipse> | <dppolyline> | <dparc>
<dpline>	**dpline** <dppt> <dppt>
<dprect>	**dprect** (**dproundr**)?
<dptextbox>	**dptxbx** ***dptxbxmar*** '{' **dptxbxtext** <para>+'}'
<dpellipse>	**dpellipse**
<dparc>	**dparc *dparcflipx**? **\dparcflipy**?
<dppolyline>	**dppolyline** (**dppolygon**)? ***dppolycount*** <dppt>+
<dppt>	***dpptx*** ***dppty***
<dphead>	***dpx*** ***dpy*** ***dpxsize*** ***dpysize***

Note that in <dpgroup> the number of <dpinfo>s is equal to the argument of **\dpcount**, while in <dppolyline> the number of <dppt>s is equal to the argument of **dppolycount.**

The following elements of the drawing-object syntax pertain specifically to callout objects:

<cotype>	**dpcotright** | **dpcotsingle** | **dpcotdouble** | **dpcottriple**
<coangle>	***dpcoa***
<coaccent>	**dpcoaccent**
<cosmartattach>	**dpcosmarta**
<cobestfit>	**dpcobestfit**
<cominusx>	**dpcominusx**
<cominusy>	**dpcominusy**
<coborder>	**dpcoborder**
<codescent>	**dpcodtop** | **dpcodcenter** | **dpcodbottom** | ***dpcodabs***

The remaining elements of the drawing object syntax are properties applied to individual drawn primitives:

<dpprops>	<lineprops>? <fillprops>? <endstylestart>? <endstyleend>? <shadow>?
<lineprops>	<linestyle> <linecolor> ***dplinew***
<linestyle>	**dplinesolid** | **dplinehollow** | **dplinedash** | **dplinedot** | **dplinedado** | **dplinedadodo**
<linecolor>	<linegray> | <linergb>
<linegray>	***dplinegray***
<linergb>	***dplinecor*** ***dplinecog*** ***dplinecob***<linepal>?
<linepal>	**\dplinepal**
<fillprops>	<fillcolorfg> <fillcolorbg> ***dpfillpat***

<fillcolorfg>	<fillfggray> | <fillfgrgb>
<fillfggray>	***dpfillfggray***
<fillfgrgb>	***dpfillfgcr*** ***dpfillfgcg*** ***dpfillfgcb***<fillfgpal>?
<fillfgpal>	**\dpfillfgpal**
<fillcolorbg>	<fillbggray> | <fillbgrgb>
<fillbggray>	***dpfillbggray***
<fillbgrgb>	***dpfillbgcr *dpfillbgcg *dpfillbgcb***<fillbgpal>?
<fillbgpal>	**\dpfillbgpal**
<endstylestart>	<arrowstartfill> ***dpastartl*** ***dpastartw***
<arrowstartfill>	**\dpastartsol** | **\dpastarthol**
<endstyleend>	<arrowendfill> ***dpaendl*** ***dpaendw***
<arrowendfill>	**\dpaendsol** | **\dpaendhol**
<shadow>	**\dpshadow** ***dpshadx*** ***dpshady***

The following table describes the control words for the drawing object group in detail. All color values are RGB values between 0-255. All distances are in twips. All other values are as indicated.

Control word	Definition
\do	Indicates a drawing object is to be inserted at this point in the character stream. This is a destination control word.
\dolock	The drawing object's anchor is locked and cannot be moved.
\dobxpage	The drawing object is page relative in the x-direction.
\dobxcolumn	The drawing object is column relative in the x-direction.
\dobxmargin	The drawing object is margin relative in the x-direction.
\dobypage	The drawing object is page relative in the y-direction.
\dobypara	The drawing object is paragraph relative in the y-direction.
\dobymargin	The drawing object is margin relative in the y-direction.
\dodhgt*N*	The drawing object is positioned at the following numeric address in the z-ordering.
Drawing Primitives	
\dpgroup	Begin group of drawing primitives.
\dpcount*N*	Number of drawing primitives in the current group.
\dpendgroup	End group of drawing primitives.
\dparc	Arc drawing primitive.
\dpcallout	Callout drawing primitive, which consists of both a polyline and a text box.
\dpellipse	Ellipse drawing primitive.
\dpline	Line drawing primitive.
\dppolygon	Polygon drawing primitive (closed polyline).
\dppolyline	Polyline drawing primitive.
\dprect	Rectangle drawing primitive.

\dptxbx Text box drawing primitive.

Position and Size

\dpx*N* X-offset of the drawing primitive from its anchor.

\dpxsize*N* X-size of the drawing primitive.

\dpy*N* Y-offset of the drawing primitive from its anchor.

\dpysize*N* Y-size of the drawing primitive.

Callouts

\dpcoa*N* Angle of callout's diagonal line is restricted to one of the following: 0, 30, 45, 60, or 90. If this control word is absent, the callout has an arbitrary angle, indicated by the coordinates of its primitives.

\dpcoaccent Accent bar on callout (vertical bar between polyline and text box).

\dpcobestfit Best fit callout (x-length of each line in callout is similar).

\dpcoborder Visible border on callout text box.

\dpcodabs*N* Absolute distance-attached polyline. ***N*** is the offset in twips from the corner that an auto-attached callout would attach to.

\dpcodbottom Bottom-attached polyline.

\dpcodcenter Center-attached polyline.

\dpcodtop Top-attached callout.

\dpcolength*N* Length of callout.

\dpcominusx Text box falls in quadrants II or III relative to polyline origin.

\dpcominusy Text box falls in quadrants III or IV relative to polyline origin.

\dpcooffset*N* Offset of callout. This is the distance between the end of the polyline and the edge of the text box.

\dpcosmarta Auto-attached callout. Polyline will attach to either the top or bottom of the text box depending on the relative quadrant.

\dpcotdouble Double line callout.

\dpcotright Right angle callout.

\dpcotsingle Single line callout.

\dpcottriple Triple line callout.

Text Boxes and Rectangles

\dptxbxmar*N* Internal margin of the text box.

\dptxbxtext Group that contains the text of the text box.

\dproundr Rectangle is a round rectangle.

Lines and Polylines

\dpptx*N* X-coordinate of the current vertex (only for lines and polylines). The coordinate order for a point must be x, y.

\dppty*N* Y-coordinate of the current vertex (only for lines and polylines). The coordinate order for a point must be x, y.

\dppolycount*N* Number of vertices in polyline drawing primitive.

Arcs

\dparcflipx This indicates that the end point of the arc is to the right of the start point. Arcs are drawn counter-clockwise.

\dparcflipy This indicates that the end point of the arc is below the start point. Arcs are drawn counter-clockwise.

Line Style

\dplinecob*N* Blue value for line color.

\dplinecog*N* Green value for line color.

\dplinecor*N* Red value for line color.

\dplinepal Render line color using the PALETTERGB macro instead of the RGB macro in Windows.

\dplinedado Dashed-dotted line style.

\dplinedadodo Dashed-dotted-dotted line style.

\dplinedash Dashed line style.

\dplinedot Dotted line style.

\dplinegray*N* Grayscale value for line color (in half-percentages).

\dplinehollow Hollow line style (no line color).

\dplinesolid Solid line style.

\dplinew*N* Thickness of line (in twips).

Arrow Style

\dpaendhol Hollow end arrow (lines only).

\dpaendl*N* Length of end arrow, relative to pen width:

1 Small

2 Medium

3 Large

\dpaendsol Solid end arrow (lines only).

\dpaendw*N* Width of end arrow, relative to pen width:

1 Small

2 Medium

3 Large

\dpastarthol Hollow start arrow (lines only).

\dpastartl*N* Length of start arrow, relative to pen width:

1 Small

2 Medium

3 Large

\dpastartsol Solid start arrow (lines only).

\dpastartw*N* Width of start arrow, relative to pen width:

1 Small

2 Medium

3 Large

Fill Pattern	
\dpfillbgcb*N*	Blue value for background fill color.
\dpfillbgcg*N*	Green value for background fill color.
\dpfillbgcr*N*	Red value for background fill color.
\dpfillbgpal	Render fill background color using the PALETTERGB macro instead of the RGB macro in Windows.
\dpfillbggray*N*	Grayscale value for background fill (in half-percentages).
\dpfillfgcb*N*	Blue value for foreground fill color.
\dpfillfgcg*N*	Green value for foreground fill color.
\dpfillfgcr*N*	Red value for foreground fill color.
\dpfillfgpal	Render fill foreground color using the PALETTERGB macro instead of the RGB macro in Windows.
\dpfillfggray*N*	Grayscale value for foreground fill (in half-percentages).
\dpfillpat*N*	Index into a list of fill patterns. See below for list.

Shadow	
\dpshadow	Current drawing primitive has a shadow.
\dpshadx*N*	X-offset of the shadow.
\dpshady*N*	Y-offset of the shadow.

The following values are available for specifying fill patterns in drawing objects with the **\dpfillpat** control word.

Value	Fill pattern
0 (zero)	Clear (no pattern)
1	Solid (100%)
2	5%
3	10%
4	20%
5	25%
6	30%
7	40%
8	50%
9	60%
10	70%
11	75%
12	80%

13	90%
14	Dark horizontal lines
15	Dark vertical lines
16	Dark left-diagonal lines (\\\)
17	Dark right-diagonal lines (///)
18	Dark grid lines
19	Dark trellis lines
20	Light horizontal lines
21	Light vertical lines
22	Light left-diagonal lines (\\\)
23	Light right-diagonal lines (///)
24	Light grid lines
25	Light trellis lines

Footnotes

The **footnote** control word introduces a footnote. Footnotes are destinations in RTF. A footnote is anchored to the character that immediately precedes the footnote destination (that is, the footnote moves with the character to which it is anchored). If automatic footnote numbering is defined, the destination can be preceded by a footnote reference character, identified by the control word **chftn**. No Microsoft product supports footnotes within headers, footers, or annotations. Placing a footnote within headers, footers, or annotations will often result in a corrupt document.

Footnotes have the following syntax.

<foot> '{*' **\footnote** <para>+ '}'

Here is an example of a destination containing footnotes:

```
\ftnbj\ftnrestart \sectd \linemod0\linex0\endnhere \pard\plain
\ri1170 \fs20 {\pu6 Mead's landmark study has been amply annotated.\chftn
{\*\footnote \pard\plain \s246 \fs20 {\up6\chftn }See Sahlins, Bateson, and
Geertz for a complete bibliography.}
It was her work in America during the Second World War, however, that forms
the basis for the paper. As others have noted, \chftn
{\*\footnote \pard\plain \s246 \fs20 {\up6\chftn}
A complete bibliography will be found at the end of this chapter.}
this period was a turning point for Margaret Mead.}
\par
```

To indicate endnotes, the following combination is emitted: **\footnote\ftnalt.** Existing readers will ignore the **\ftnalt** control word and treat everything as a footnote.

For other control words relating to footnotes, see the sections titled "Document Formatting Properties" (page 17), "Error: Reference source not found" (page 20), and "Special Characters" (page 37) in this Application Note.

Annotations

RTF annotations have two parts; the author ID (introduced by the control word **atnid**) and the annotation text (introduced by the control word **annotation**); there is no group enclosing both parts. No Microsoft product supports annotations within headers, footers, or footnotes. Placing an annotation within headers, footers, or footnotes will often result in a corrupt document. Each part of the annotation is an RTF destination. Annotations are anchored to the character that immediately precedes the annotation.

If an annotation is associated with an annotation bookmark, the following two destination control words precede and follow the bookmark. The alphanumeric string ***N***, such as a long integer, represents the bookmark name.

<atrfstart>	'{*' **\atrfstart *N*** '}'
<atrfend>	'{*' **\atrfend *N*** '}'

Annotations have the following syntax:

<annot>	<annotid> <atnauthor> <atntime>? **\chatn** <atnicn>? <annotdef>
<annotid>	'{*' **\atnid** #PCDATA '}'
<atnauthor>	'{*' **\atnauthor** #PCDATA '}'
<annotdef>	'{*' **\annotation** <atnref> <para>+ '}'
<atnref>	'{*' **\atnref *N*** '}'
<atntime>	'{*' **\atntime** <time> '}'
<atnicn>	'{*' **\atnicn** <pict> '}'

An example of annotation text follows:

```
An example of a paradigm might be Newtonian physics or
Darwinian biology.{\v\fs16 {\atnid bz}\chatn{\*\annotation
\pard\plain \s224 \fs20 {\field{\fldinst page \\#'"Page:
'#'\line'"}{\fldrslt}}{\fs16 \chatn }
How about some examples that deal with social science?
That's what this paper is about.}}
```

Annotations may have optional time stamps (contained in the **atntime** destination) or icons (contained in the **atnicn** destination).

Fields

The **field** control word introduces a field destination, which contains the text of Word for Windows fields.

Fields have the following syntax:

<field>	'{' **\field** <fieldmod>? <fieldinst> <fieldrslt> '}'
<fieldmod>	**\flddirty**? & **\fldedit**? & **\fldlock**? & **\fldpriv**?
<fieldinst>	'{*' **\fldinst** <char>+ <fldalt>? '}'
<fldalt>	**\fldalt**
<fieldrslt>	'{*' **\fldrslt** <para>+ '}'

There are several control words that alter the interpretation of the field. These control words are listed in the following table.

Control word	Meaning
\flddirty	A formatting change has been made to the field result since the field was last updated.
\fldedit	Text has been added to, or removed from, the field result since the field was last updated.
\fldlock	Field is locked and cannot be updated.
\fldpriv	Result is not in a form suitable for display (for example, binary data used by fields whose result is a picture).

Two subdestinations are required within the **field** destination. They must be enclosed in braces ({ }) and begin with the following control words.

Control word	Meaning
\fldinst	Field instructions. This is a destination control word.
\fldrslt	Most recent calculated result of the field. This is a destination control word.

If the instruction for a field contains a filename, then the **cpg** control can be used to define the character set of the filename. See “Code Page Support” on page 11 of this Application Note for details.

The **fldrslt** control word should be included even if no result has been calculated because most readers (even those readers that do not recognize fields) can generally include the value of the **fldrslt** destination in the document.

An example of some field text follows:

```
{\field\fldedit{\fldinst author}{\fldrslt Joe Smith}}\par\pard
{\field{\fldinst time \\@ "h:mm AM/PM"}{\fldrslt 8:12 AM}}
```

You can use the **\fldalt** control word to specify that the given field reference is to an endnote. For example, the following field in RTF is a reference to a footnote:

```
{\ field{\ *\ fldinst NOTEREF _RefNumber } {\ fldrslt 1}}
```

The following is an example of a reference to an endnote:

```
{\ field{\ *\ fldinst NOTEREF _RefNumber \ fldalt } {\ fldrslt I}}
```

If the specified field is a form field, the ***\datafield** destination appears as a part of <char> and contains the binary data of a form field instruction. For example:

```
{\ field{\*\fldinst {\*\bkmkstart Text1} FORMTEXT {{\*\datafield
00000000000000000554657874310008476565207768697a000000000000000000000000}}}{\fldrslt Default Result}}
{\*\bkmkend Text1}
```

Note that the **\datafield** destination requires the ***** prefix.

Index Entries

The **xe** control word introduces an index entry. Index entries in RTF are destinations. An index entry has the following syntax:

<idx>	'{' **\xe** (***xef***? & **bxe**? & **ixe**?) <char>+ (<txe> | <rxe>)? '}'
<txe>	'{' **\txe** <char>+ '}'
<rxe>	'{' **\rxe** #PCDATA '}'

If the text of the index entry is not formatted as hidden text with the **v** control word, the text is put into the document as well as into the index. For more information on the **v** control word, see "Character Formatting Properties" on page 33 of this Application Note. Similarly, the text of the **txe** subdestination, described later in this section, becomes part of the document if it is not formatted as hidden text.

The following control words may also be used.

Control word	Meaning
\xef*N*	Allows multiple indexes within the same document. ***N*** is an integer that corresponds to the ASCII value of a letter between A and Z.
\bxe	Formats the page number or cross-reference in bold.
\ixe	Formats the page number or cross-reference in italic.
\txe *Text*	Text argument to be used instead of a page number. This is a destination control word.

\rxe ***BookmarkName***	Text argument is a bookmark for the range of page numbers. This is a destination control word.

Table of Contents Entries

The **tc** control word introduces a table of contents entry, which can be used to build the actual table of contents. The **\tcn** control word marks a table of contents entry that will not have a page number associated with it; this is used in place of **\tc** for such entries. Table of contents entries are destinations, and they have the following syntax:

<toc> '{' **\tc | \tcn** (***tcf***? & ***tcl***?) <char>+ '}'

As with index entries, text that is not formatted as hidden with the **v** character-formatting control word is put into the document. The following control words can also be used in this destination.

Control word	Meaning
\tcf*N*	Type of table being compiled; ***N*** is mapped by existing Microsoft software to a letter between A and Z (the default is 67, which maps to C, used for tables of contents).
\tcl*N*	Level number (the default is 1).

Bidirectional Language Support

RTF supports bidirectional writing orders for languages such as Arabic. The controls are described below (as well as in the appropriate sections throughout this Application Note). Also refer to the associated character properties defined in “” on page 36 of this Application Note.

All the control words relating to bidirectional language support are repeated here for convenience.

Control word	Meaning
\rtlch	The character data following this control word will be treated as a right-to-left run.
\ltrch	The character data following this control word will be treated as a left-to-right run (the default).
\rtlmark	The following characters should be displayed from right to left.
\ltrmark	The following characters should be displayed from left to right.
\rtlpar	Text in this paragraph will be displayed with right-to-left precedence
\ltrpar	Text in this paragraph will be displayed with left-to-right precedence. This is the default.
\rtlrow	Cells in this table row will have right-to-left precedence.
\ltrrow	Cells in this table row will have left-to-right precedence. This is the default.
\rtlsect	This section will thread columns from right to left.
\ltrsect	This section will thread columns from left to right. This is the default.
\rtldoc	Text in this document will be displayed from right to left unless overridden by a more specific control.
\ltrdoc	Text in this document will be displayed from left to right unless overridden by a more specific control. This is the default.
\zwj	Zero-width joiner. This is used for ligating characters.
\zwnj	Zero-width nonjoiner. This is used for unligating characters.

APPENDIX A: SAMPLE RTF READER APPLICATION

The GC0165 disk included with this Application Note contains the sample RTF reader program RTFREADR.EXE, which will help you create an RTF reader for your own application when used in conjunction with the Microsoft Rich Text Format Specification and the information below.

> ***Note:*** *The sample RTF reader is not a for-sale product, and Microsoft does not provide technical or any other type of support for the sample RTF reader code or the RTF specification.*

If this shipment has arrived in unsatisfactory condition, please call Microsoft Product Support Services (PSS). In the United States, call (206) 462-WORD (9673) between 6:00 A.M. and 6:00 P.M. Pacific time. Outside the United States, contact the Microsoft subsidiary for your area. To locate your subsidiary, call Microsoft International Customer Service at (206) 936-8661.

How to Write an RTF Reader

There are three basic things that an RTF reader must do:

1.Separate text from RTF controls

2.Parse an RTF control

3.Dispatch an RTF control

Separating text from RTF controls is relatively simple, as all RTF controls begin with a backslash. Therefore, any incoming character that is not a backslash is text and will be handled as text. (Of course, what one *does* with that text may be relatively complicated.)

Parsing an RTF control is also relatively simple. An RTF control is either (a) a sequence of alphabetic characters followed by an optional numeric parameter, or (b) a single non-alphanumeric character.

Dispatching an RTF control, on the other hand, is relatively complicated. A recursive-descent parser tends to be overly strict because RTF is intentionally vague about the order of various properties relative to one another. However, whatever method you use to dispatch an RTF control, your reader should do the following:

- **Ignore keywords you don't understand.**

  Many readers crash when they come across an unknown RTF control. Because Microsoft is continually adding new RTF controls, this limits an RTF reader to working with the RTF from one particular product (usually some version of Word for Windows).

- **Always understand *.**

  One of the most important things an RTF reader can do is to understand the * control. This control introduces a destination that is not part of the document. It tells the RTF reader that if the reader does not understand the next control word, then it should skip the entire enclosing group. If your reader follows this rule and the one above, your reader will be able to cope with any future change to RTF short of a complete rewrite.

- **Remember that binary data can occur when you're skipping RTF.**

  A simple way to skip a group in RTF is to keep a running count of the opening curly braces that the reader has encountered in the RTF stream. When the reader sees an opening curly brace, it increments the count; when the reader sees a closing curly brace, it decrements the count. When the count becomes negative, the end of the group has been found. Unfortunately, this doesn't work when the RTF file contains a **\bin** control; the reader must explicitly check each control word found to see if it's a **\bin** control, and—if a **\bin** control is found—skip that many bytes before resuming its scanning for curly braces.

A Sample RTF Reader Implementation

The Microsoft Word Processing Conversions group uses a table-driven approach to reading RTF. This approach allows the most flexibility in reading RTF, with the corresponding problem that it's difficult to detect incorrect RTF. An RTF reader that is based on this approach is presented below. This reader works exactly as described in the RTF specification and uses the principles of operation described in the RTF specification. This reader is designed

to be simple to understand but is not intended to be very efficient. This RTF reader also implements the three design principles listed in the previous section.

The RTF reader consists of four files:

- RTFDECL.H, which contains the prototypes for all the functions in the RTF reader
- RTFTYPE.H, which contains the types used in the RTF reader
- RTFREADR.C, which contains the main program, the main loop of the RTF reader, and the RTF control parser
- RTFACTN.C, which contains the dispatch routines for the RTF reader

RTFDECL.H and RTFREADR.C

RTFDECL.H is straightforward and requires little explanation.

RTFREADR.C is also reasonably straightforward; the function ecRtfParse separates text from RTF controls and handles text, and the function ecParseRtfKeyword parses an RTF control and also collects any parameter that follows the RTF control.

RTFTYPE.H

RTFTYPE.H begins by declaring a sample set of character, paragraph, section, and document properties. These structures are present to demonstrate how the dispatch routines can modify any particular property and are not actually used to format text.

For example, the following enumeration describes which destination text should be routed to:

```
typedef enum { rdsNorm, rdsSkip } RDS;
```

Because this is just a sample RTF reader, there are only two destinations; a more complicated reader would add an entry to this enumeration for each destination supported (for example—headers, footnotes, endnotes, annotations, bookmarks, and pictures).

The following enumeration describes the internal state of the RTF parser:

```
typedef enum { risNorm, risBin, risHex } RIS;
```

This is entirely separate from the state of the dispatch routines and the destination state; other RTF readers may not necessarily have anything similar to this.

The following structure encapsulates the state that must be saved at a group start and restored at a group end:

```
typedef struct save
{
struct save *pNext;
CHP chp;
PAP pap;
SEP sep;
DOP dop;
RDS rds;
RIS ris;
} SAVE;
```

The following enumeration describes a set of classes for RTF controls:

```
typedef enum {kwdChar, kwdDest, kwdProp, kwdSpec} KWD;
```

Use kwdChar for controls that represent special characters (such as **\-**, **\{**, or **\}**).

Use kwdDest for controls that introduce RTF destinations.

Use kwdProp for controls that modify some sort of property.

Use kwdSpec for controls that need to run some specialized code.

The following enumeration defines the number of PROP structures (described below) that will be used. There will typically be an iprop for every field in the character, paragraph, section, and document properties.

```
typedef enum {ipropBold, ipropItalic, ipropUnderline, ipropLeftInd,
ipropRightInd, ipropFirstInd, ipropCols, ipropPgnX, ipropPgnY,
ipropXaPage, ipropYaPage, ipropXaLeft, ipropXaRight,
ipropYaTop, ipropYaBottom, ipropPgnStart, ipropSbk,
ipropPgnFormat, ipropFacingp, ipropLandscape, ipropJust,
ipropPard, ipropPlain,
ipropMax} IPROP;
```

The following structure is a very compact way to describe how to locate the address of a particular value in one of the property structures:

```
typedef enum {actnSpec, actnByte, actnWord} ACTN;
typedef enum {propChp, propPap, propSep, propDop} PROPTYPE;

typedef struct propmod
{
ACTN actn;
PROPTYPE prop;
int offset;
} PROP;
```

The `actn` field describes the width of the value being described: if the value is a byte, then `actn` is `actnByte`; if the value is a word, then `actn` is `actnWord`; if the value is neither a byte nor a word, then you can use `actnSpec` to indicate that some C code needs to be run to set the value. The `prop` field indicates which property structure is being described; `propChp` indicates that the value is located within the CHP structure; `propPap` indicates that the value is located within the PAP structure, and so on. Finally, the offset field contains the offset of the value from the start of the structure. The offsetof() macro is usually used to initialize this field.

The following structure describes how to parse a particular RTF control:

```
typedef enum {ipfnBin, ipfnHex, ipfnSkipDest } IPFN;
typedef enum {idestPict, idestSkip } IDEST;

typedef struct symbol
{
char *szKeyword;
int dflt;
bool fPassDflt;
KWD kwd;
int idx;
} SYM;
```

`szKeyword` points to the RTF control being described; `kwd` describes the class of the particular RTF control (described above); `dflt` is the default value for this control, and `fPassDflt` should be nonzero if the value in `dflt` should be passed to the dispatch routine. (`fPassDflt` is only nonzero for keywords that normally set a particular value. For example, the various section break controls typically have nonzero `fPassDflt` controls, but controls that take parameters should not.)

`Idx` is a generalized index; its use depends on the `kwd` being used for this control.

- If `kwd` is kwdChar, then `idx` is the character that should be output.
- If `kwd` is kwdDest, then `idx` is the `idest` for the new destination.
- If `kwd` is kwdProp, then `idx` is the `iprop` for the appropriate property.
- If `kwd` is kwdSpec, then `idx` is an `ipfn` for the appropriate function.

With this structure, it is very simple to dispatch an RTF keyword. Once the reader isolates the RTF keyword and its (possibly associated) value, the reader then searches an array of SYM structures to find the RTF keyword. If the keyword is not found, the reader ignores it, unless the previous control was *****, in which case the reader must scan past an entire group.

If the keyword is found, the reader then uses the kwd value from the SYM structure to determine what to do. This is, in fact, exactly what the function ecTranslateKeyword in the file RTFACTN.C does.

The RTFACTN.C File

RTFACTN.C contains the tables describing the properties and keywords, and the routines to evaluate properties (ecApplyPropChange) and to dispatch keywords (ecTranslateKeyword).

The tables are the keys to understanding the RTF dispatch routines. The following are some sample entries from both tables, along with a brief explanation of each entry.

The Property Table. This table must have an entry for every iprop.

```
actnByte,    propChp,     offsetof(CHP, fBold),         // ipropBold
```

This property says that the ipropBold property is a byte parameter bound to chp.fBold.

```
actnWord,    propPap,     offsetof(PAP, xaRight),      // ipropRightInd
```

This property says that ipropRightInd is a word parameter bound to pap.xaRight.

```
actnWord,    propSep,     offsetof(SEP, cCols),         // ipropCols
```

This property says that ipropCols is a word parameter bound to sep.cCols.

```
actnSpec,    propChp,     0,                                  // ipropPlain
```

This property says that ipropPlain is a special parameter. Instead of directly evaluating it, ecApplyPropChange will run some custom C code to apply a property change.

The Keyword Table.

```
"b",          1,        fFalse,       kwdProp,     ipropBold,
```

This structure says that the control **\b** sets the ipropBold property. Because fPassDflt is false, the reader only uses the default value if the control does not have a parameter. If no parameter is provided, the reader uses a value of 1.

```
"sbknone",   sbkNon, fTrue,        kwdProp,     ipropSbk,
```

This entry says that the control **\sbknone** sets the ipropSbk property. Since fPassDflt is true, the reader always uses the default value of sbkNon, even if the control has a parameter.

```
"par",        0,        fFalse,       kwdChar,     0x0a,
```

This entry says that the control **\par** is equivalent to a 0x0a (linefeed) character.

```
"tab",        0,        fFalse,       kwdChar,     0x09,
```

This entry says that the control **\tab** is equivalent to a 0x09 (tab) character.

```
"bin",        0,        fFalse,       kwdSpec,     ipfnBin,
```

This entry says that the control **\bin** should run some C code. The particular piece of C code can be located by the ipfnBin parameter.

```
"fonttbl",   0,        fFalse,       kwdDest,     idestSkip,
```

This entry says that the control **\fonttbl** should change to the destination idestSkip.

Notes on Implementing Other RTF Features

The table-driven approach to dispatching RTF controls used by the sample converter does not implement any syntax checking. For most controls, this is not a problem; a control simply modifies the appropriate property. However, some controls, such as those for tabs and borders, are dependent on other control words either before or after the current control word.

There are some standard techniques for handling these features:

Tabs and Other Control Sequences Terminating in a Fixed Control

The best way to implement these types of control sequences is to have a global structure that represents the current state of the tab descriptor (or other entity). As the modifiers come in, they modify the various fields of the global structure; when the fixed control at the end of the sequence is dispatched, it adds the entire descriptor and reinitializes the global variable.

Borders and Other Control Sequences Beginning with a Fixed Control

The best way to implement these types of control sequences is to have a global pointer that is initialized when the fixed control is dispatched; the controls that modify the fixed control then modify fields pointed to by the control.

Other Problem Areas in RTF

Style Sheets

Style sheets can be handled as destinations; however, styles have default values, just as every other control does; RTF readers should be sure to handle a missing style control as the default style value (that is, 0).

Property Changes

Some RTF readers use various bits of RTF syntax to mark property changes. In particular, they assume that property changes will occur only after a group start, which is not correct. Because there is a variety of ways to represent identical property changes in RTF, RTF readers should look at the changes in the properties and not at any particular way of representing a property change. In particular, properties can be changed explicitly with a control word or implicitly at the end of a group. For example, these three sequences of RTF have exactly the same semantics, and should be translated identically:

- `{\b bold \i Bold Italic \i0 Bold again}`
- `{\b bold {\i Bold Italic }Bold again}`
- `{\b bold \i Bold Italic \plain\b Bold again}`

Fields

All versions of Microsoft Word for Windows and version 6.0 of Microsoft Word for the Macintosh have fields. If you're writing an RTF reader and expect to do anything with fields, keep the following notes in mind:

- Field instructions may have arbitrary amounts of character formatting and arbitrarily nested groups. While the groups will be properly nested within the field instructions, you may be inside an arbitrary number of groups by the time you know which field you working with. If you then expect to be able to skip to the end of the field instructions, you'll have to know how many groups have started so that you can skip to the end properly.
- Some fields, the INCLUDE field in particular, can have section breaks in the field results. If this occurs, then the text after the end of the field does not have the same section properties as the text at the start of the field; the section properties must not be restored when the field results contain section breaks.

Tables

Tables are probably the trickiest part of RTF to read and write correctly. Because of the way Microsoft word processors implement tables, and the table-driven approach of many Microsoft RTF readers, it is very easy to

write tables in RTF that will crash Microsoft word processors when you try to read the RTF. Here are some guidelines to reduce problems with tables in RTF:

- Place the entire table definition before any paragraph properties, including **\pard**.
- Make sure the number of cells in the RTF matches the number of cell definitions.
- Some controls must be the same in all paragraphs in a row. In particular, all paragraphs in a row must have the same positioning controls, and all paragraphs in a row must have **\intbl** specified.
- Do not use the **\sbys** control inside a table. **\sbys** is a holdover from Word for MS-DOS and early versions of Word for the Macintosh. Word for Windows and current versions of Word for the Macintosh translate **\sbys** as a table. Because Word for Windows and Word for the Macintosh do not support nested tables, these products will probably crash if you specify **\sbys** in a table.
- Cell definitions starting before the left margin of the paper begins (that is, the parameter plus the left margin is negative) are always in error.
- Even though nested tables are not explicitly defined in RTF, and Word for Windows and Word for the Macintosh do not support nested tables, you must still save table properties when changing destinations because tables can be nested inside other destinations—that is, you can have a table that contains a footnote or an annotation, and the footnote or annotation can contain another table.

Appendix A-1: Listings

RTFDECL.H

```
// RTF parser declarations

int ecRtfParse(FILE *fp);
int ecPushRtfState(void);
int ecPopRtfState(void);
int ecParseRtfKeyword(FILE *fp);
int ecParseChar(int c);
int ecTranslateKeyword(char *szKeyword, int param, bool fParam);
int ecPrintChar(int ch);
int ecEndGroupAction(RDS rds);
int ecApplyPropChange(IPROP iprop, int val);
int ecChangeDest(IDEST idest);
int ecParseSpecialKeyword(IPFN ipfn);
int ecParseSpecialProperty(IPROP iprop, int val);
int ecParseHexByte(void);

// RTF variable declarations

extern int cGroup;
extern RDS rds;
extern RIS ris;

extern CHP chp;
extern PAP pap;
extern SEP sep;
extern DOP dop;

extern SAVE *psave;
extern long cbBin;
extern long lParam;
extern bool fSkipDestIfUnk;
extern FILE *fpIn;

// RTF parser error codes

#define ecOK 0                             // Everything's fine!
#define ecStackUnderflow     1         // Unmatched '}'
#define ecStackOverflow      2         // Too many '{' -- memory exhausted
#define ecUnmatchedBrace     3         // RTF ended during an open group.
#define ecInvalidHex          4         // invalid hex character found in data
#define ecBadTable            5         // RTF table (sym or prop) invalid
#define ecAssertion           6         // Assertion failure
#define ecEndOfFile           7         // End of file reached while reading RTF
```

RTFTYPE.H

```
typedef char bool;
#define fTrue 1
#define fFalse 0

typedef struct char_prop
{
     char fBold;
     char fUnderline;
     char fItalic;
} CHP;                        // CHaracter Properties

typedef enum {justL, justR, justC, justF } JUST;
typedef struct para_prop
{
     int xaLeft;                      // left indent in twips
     int xaRight;                     // right indent in twips
     int xaFirst;                     // first line indent in twips
```

```
     JUST just;                         // justification
} PAP;                        // PAragraph Properties

typedef enum {sbkNon, sbkCol, sbkEvn, sbkOdd, sbkPg} SBK;
typedef enum {pgDec, pgURom, pgLRom, pgULtr, pgLLtr} PGN;
typedef struct sect_prop
{
     int cCols;                         // number of columns
     SBK sbk;                           // section break type
     int xaPgn;                         // x position of page number in twips
     int yaPgn;                         // y position of page number in twips
     PGN pgnFormat;                    // how the page number is formatted
} SEP;                        // SEction Properties

typedef struct doc_prop
{
     int xaPage;                        // page width in twips
     int yaPage;                        // page height in twips
     int xaLeft;                        // left margin in twips
     int yaTop;                         // top margin in twips
     int xaRight;                      // right margin in twips
     int yaBottom;                     // bottom margin in twips
     int pgnStart;                     // starting page number in twips
     char fFacingp;                    // facing pages enabled?
     char fLandscape;                 // landscape or portrait??
} DOP;                        // DOcument Properties

typedef enum { rdsNorm, rdsSkip } RDS;                  // Rtf Destination State
typedef enum { risNorm, risBin, risHex } RIS;         // Rtf Internal State

typedef struct save                 // property save structure
{
     struct save *pNext;            // next save
     CHP chp;
     PAP pap;
     SEP sep;
     DOP dop;
     RDS rds;
     RIS ris;
} SAVE;

// What types of properties are there?
typedef enum {ipropBold, ipropItalic, ipropUnderline, ipropLeftInd,
                  ipropRightInd, ipropFirstInd, ipropCols, ipropPgnX,
                  ipropPgnY, ipropXaPage, ipropYaPage, ipropXaLeft,
                  ipropXaRight, ipropYaTop, ipropYaBottom, ipropPgnStart,
                  ipropSbk, ipropPgnFormat, ipropFacingp, ipropLandscape,
                  ipropJust, ipropPard, ipropPlain, ipropSectd,
                  ipropMax } IPROP;

typedef enum {actnSpec, actnByte, actnWord} ACTN;
typedef enum {propChp, propPap, propSep, propDop} PROPTYPE;

typedef struct propmod
{
     ACTN actn;                  // size of value
     PROPTYPE prop;             // structure containing value
     int   offset;               // offset of value from base of structure
} PROP;

typedef enum {ipfnBin, ipfnHex, ipfnSkipDest } IPFN;
typedef enum {idestPict, idestSkip } IDEST;
typedef enum {kwdChar, kwdDest, kwdProp, kwdSpec} KWD;

typedef struct symbol
{
     char *szKeyword;          // RTF keyword
     int   dflt;                 // default value to use
     bool fPassDflt;           // true to use default value from this table
     KWD   kwd;                  // base action to take
     int   idx;                  // index into property table if kwd == kwdProp
                                 // index into destination table if kwd == kwdDest
                                 // character to print if kwd == kwdChar
```

```
} SYM;
```

RTFREADR.C

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "rtftype.h"
#include "rtfdecl.h"

int cGroup;
bool fSkipDestIfUnk;
long cbBin;
long lParam;

RDS rds;
RIS ris;

CHP chp;
PAP pap;
SEP sep;
DOP dop;

SAVE *psave;
FILE *fpIn;

//
// %%Function: main
//
// Main loop. Initialize and parse RTF.
//
main(int argc, char *argv[])
{
     FILE *fp;
     int ec;

     fp = fpIn = fopen("test.rtf", "r");
     if (!fp)
     {
          printf ("Can't open test file!\n");
          return 1;
     }
     if ((ec = ecRtfParse(fp)) != ecOK)
          printf("error %d parsing rtf\n", ec);
     else
          printf("Parsed RTF file OK\n");
     fclose(fp);
     return 0;
}

//
// %%Function: ecRtfParse
//
// Step 1:
// Isolate RTF keywords and send them to ecParseRtfKeyword;
// Push and pop state at the start and end of RTF groups;
// Send text to ecParseChar for further processing.
//

int
ecRtfParse(FILE *fp)
{
     int ch;
     int ec;
     int cNibble = 2;
     int b = 0;
     while ((ch = getc(fp)) != EOF)
     {
          if (cGroup < 0)
               return ecStackUnderflow;
          if (ris == risBin)                          // if we're parsing binary data, handle it
directly
          {
```

```
                if ((ec = ecParseChar(ch)) != ecOK)
                    return ec;
            }
            else
            {
                switch (ch)
                {
                case '{':
                    if ((ec = ecPushRtfState()) != ecOK)
                        return ec;
                    break;
                case '}':
                    if ((ec = ecPopRtfState()) != ecOK)
                        return ec;
                    break;
                case '\\':
                    if ((ec = ecParseRtfKeyword(fp)) != ecOK)
                        return ec;
                    break;
                case 0x0d:
                case 0x0a:              // cr and lf are noise characters...
                    break;
                default:
                    if (ris == risNorm)
                    {
                        if ((ec = ecParseChar(ch)) != ecOK)
                            return ec;
                    }
                    else
                    {                   // parsing hex data
                        if (ris != risHex)
                            return ecAssertion;
                        b = b << 4;
                        if (isdigit(ch))
                            b += (char) ch - '0';
                        else
                        {
                            if (islower(ch))
                            {
                                if (ch < 'a' || ch > 'f')
                                    return ecInvalidHex;
                                b += (char) ch - 'a';
                            }
                            else
                            {
                                if (ch < 'A' || ch > 'F')
                                    return ecInvalidHex;
                                b += (char) ch - 'A';
                            }
                        }
                        cNibble--;
                        if (!cNibble)
                        {
                            if ((ec = ecParseChar(ch)) != ecOK)
                                return ec;
                            cNibble = 2;
                            b = 0;
ris = risNorm;
                        }
                    }                       // end else (ris != risNorm)
                    break;
                }           // switch
            }               // else (ris != risBin)
        }                   // while
        if (cGroup < 0)
            return ecStackUnderflow;
        if (cGroup > 0)
            return ecUnmatchedBrace;
        return ecOK;
}

//
// %%Function: ecPushRtfState
```

```
//
// Save relevant info on a linked list of SAVE structures.
//

int
ecPushRtfState(void)
{
     SAVE *psaveNew = malloc(sizeof(SAVE));
     if (!psaveNew)
          return ecStackOverflow;

     psaveNew -> pNext = psave;
     psaveNew -> chp = chp;
     psaveNew -> pap = pap;
     psaveNew -> sep = sep;
     psaveNew -> dop = dop;
     psaveNew -> rds = rds;
     psaveNew -> ris = ris;
     ris = risNorm;
     psave = psaveNew;
     cGroup++;
     return ecOK;
}

//
// %%Function: ecPopRtfState
//
// If we're ending a destination (that is, the destination is changing),
// call ecEndGroupAction.
// Always restore relevant info from the top of the SAVE list.
//

int
ecPopRtfState(void)
{
     SAVE *psaveOld;
     int ec;

     if (!psave)
          return ecStackUnderflow;

     if (rds != psave->rds)
     {
          if ((ec = ecEndGroupAction(rds)) != ecOK)
               return ec;
     }
     chp = psave->chp;
     pap = psave->pap;
     sep = psave->sep;
     dop = psave->dop;
     rds = psave->rds;
     ris = psave->ris;

     psaveOld = psave;
     psave = psave->pNext;
     cGroup--;
     free(psaveOld);
     return ecOK;
}

//
// %%Function: ecParseRtfKeyword
//
// Step 2:
// get a control word (and its associated value) and
// call ecTranslateKeyword to dispatch the control.
//

int
ecParseRtfKeyword(FILE *fp)
{
     int ch;
     char fParam = fFalse;
```

```
     char fNeg = fFalse;
     int param = 0;
     char *pch;
     char szKeyword[30];
     char szParameter[20];

     szKeyword[0] = '\0';
     szParameter[0] = '\0';
     if ((ch = getc(fp)) == EOF)
          return ecEndOfFile;
     if (!isalpha(ch))              // a control symbol; no delimiter.
     {
          szKeyword[0] = (char) ch;
          szKeyword[1] = '\0';
          return ecTranslateKeyword(szKeyword, 0, fParam);
     }
     for (pch = szKeyword; isalpha(ch); ch = getc(fp))
          *pch++ = (char) ch;
     *pch = '\0';
     if (ch == '-')
     {
          fNeg   = fTrue;
          if ((ch = getc(fp)) == EOF)
               return ecEndOfFile;
     }
     if (isdigit(ch))
     {
          fParam = fTrue;           // a digit after the control means we have a parameter
          for (pch = szParameter; isdigit(ch); ch = getc(fp))
               *pch++ = (char) ch;
          *pch = '\0';
          param = atoi(szParameter);
          if (fNeg)
               param = -param;
          lParam = atol(szParameter);
          if (fNeg)
               param = -param;
     }
     if (ch != ' ')
          ungetc(ch, fp);
     return ecTranslateKeyword(szKeyword, param, fParam);
}

//
// %%Function: ecParseChar
//
// Route the character to the appropriate destination stream.
//

int
ecParseChar(int ch)
{
     if (ris == risBin && --cbBin <= 0)
          ris = risNorm;
     switch (rds)
     {
     case rdsSkip:
          // Toss this character.
          return ecOK;
     case rdsNorm:
          // Output a character. Properties are valid at this point.
          return ecPrintChar(ch);
     default:
     // handle other destinations....
          return ecOK;
     }
}

//
// %%Function: ecPrintChar
//
// Send a character to the output file.
//
```

```
int
ecPrintChar(int ch)
{
     // unfortunately, we don't do a whole lot here as far as layout goes...
     putchar(ch);
     return ecOK;
}
```

RTFACTN.C

```
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include <ctype.h>
#include "rtftype.h"
#include "rtfdecl.h"

// RTF parser tables

// Property descriptions
PROP rgprop [ipropMax] = {
     actnByte,    propChp,     offsetof(CHP, fBold),         // ipropBold
     actnByte,    propChp,     offsetof(CHP, fItalic),       // ipropItalic
     actnByte,    propChp,     offsetof(CHP, fUnderline),    // ipropUnderline
     actnWord,    propPap,     offsetof(PAP, xaLeft),        // ipropLeftInd
     actnWord,    propPap,     offsetof(PAP, xaRight),       // ipropRightInd
     actnWord,    propPap,     offsetof(PAP, xaFirst),       // ipropFirstInd
     actnWord,    propSep,     offsetof(SEP, cCols),         // ipropCols
     actnWord,    propSep,     offsetof(SEP, xaPgn),         // ipropPgnX
     actnWord,    propSep,     offsetof(SEP, yaPgn),         // ipropPgnY
     actnWord,    propDop,     offsetof(DOP, xaPage),        // ipropXaPage
     actnWord,    propDop,     offsetof(DOP, yaPage),        // ipropYaPage
     actnWord,    propDop,     offsetof(DOP, xaLeft),        // ipropXaLeft
     actnWord,    propDop,     offsetof(DOP, xaRight),       // ipropXaRight
     actnWord,    propDop,     offsetof(DOP, yaTop),         // ipropYaTop
     actnWord,    propDop,     offsetof(DOP, yaBottom),      // ipropYaBottom
     actnWord,    propDop,     offsetof(DOP, pgnStart),      // ipropPgnStart
     actnByte,    propSep,     offsetof(SEP, sbk),           // ipropSbk
     actnByte,    propSep,     offsetof(SEP, pgnFormat),     // ipropPgnFormat
     actnByte,    propDop,     offsetof(DOP, fFacingp),      // ipropFacingp
     actnByte,    propDop,     offsetof(DOP, fLandscape),    // ipropLandscape
     actnByte,    propPap,     offsetof(PAP, just),          // ipropJust
     actnSpec,    propPap,     0,                                  // ipropPard
     actnSpec,    propChp,     0,                                  // ipropPlain
     actnSpec,    propSep,     0,                                  // ipropSectd
};

// Keyword descriptions
SYM rgsymRtf[] = {
//   keyword      dflt     fPassDflt    kwd           idx
     "b",          1,        fFalse,      kwdProp,     ipropBold,
     "u",          1,        fFalse,      kwdProp,     ipropUnderline,
     "i",          1,        fFalse,      kwdProp,     ipropItalic,
     "li",         0,        fFalse,      kwdProp,     ipropLeftInd,
     "ri",         0,        fFalse,      kwdProp,     ipropRightInd,
     "fi",         0,        fFalse,      kwdProp,     ipropFirstInd,
     "cols",       1,        fFalse,      kwdProp,     ipropCols,
     "sbknone",   sbkNon, fTrue,        kwdProp,     ipropSbk,
     "sbkcol",    sbkCol, fTrue,        kwdProp,     ipropSbk,
     "sbkeven",   sbkEvn, fTrue,        kwdProp,     ipropSbk,
     "sbkodd",    sbkOdd, fTrue,        kwdProp,     ipropSbk,
     "sbkpage",   sbkPg,   fTrue,       kwdProp,     ipropSbk,
     "pgnx",       0,        fFalse,      kwdProp,     ipropPgnX,
     "pgny",       0,        fFalse,      kwdProp,     ipropPgnY,
     "pgndec",    pgDec,   fTrue,       kwdProp,     ipropPgnFormat,
     "pgnucrm",   pgURom, fTrue,        kwdProp,     ipropPgnFormat,
     "pgnlcrm",   pgLRom, fTrue,        kwdProp,     ipropPgnFormat,
     "pgnucltr", pgULtr, fTrue,       kwdProp,     ipropPgnFormat,
     "pgnlcltr", pgLLtr, fTrue,       kwdProp,     ipropPgnFormat,
     "qc",         justC,   fTrue,       kwdProp,     ipropJust,
     "ql",         justL,   fTrue,       kwdProp,     ipropJust,
     "qr",         justR,   fTrue,       kwdProp,     ipropJust,
     "qj",         justF,   fTrue,       kwdProp,     ipropJust,
     "paperw",    12240,   fFalse,      kwdProp,     ipropXaPage,
     "paperh",    15480,   fFalse,      kwdProp,     ipropYaPage,
     "margl",     1800,    fFalse,      kwdProp,     ipropXaLeft,
     "margr",     1800,    fFalse,      kwdProp,     ipropXaRight,
     "margt",     1440,    fFalse,      kwdProp,     ipropYaTop,
     "margb",     1440,    fFalse,      kwdProp,     ipropYaBottom,
```

```
      "pgnstart", 1,        fTrue,        kwdProp,     ipropPgnStart,
      "facingp",   1,       fTrue,        kwdProp,     ipropFacingp,
      "landscape",1,        fTrue,        kwdProp,     ipropLandscape,
      "par",        0,       fFalse,       kwdChar,     0x0a,
      "\0x0a",     0,       fFalse,       kwdChar,     0x0a,
      "\0x0d",     0,       fFalse,       kwdChar,     0x0a,
      "tab",        0,       fFalse,       kwdChar,     0x09,
      "ldblquote",0,        fFalse,       kwdChar,     '"',
      "rdblquote",0,        fFalse,       kwdChar,     '"',
      "bin",        0,       fFalse,       kwdSpec,     ipfnBin,
      "*",          0,       fFalse,       kwdSpec,     ipfnSkipDest,
      "'",          0,       fFalse,       kwdSpec,     ipfnHex,
      "author",    0,       fFalse,       kwdDest,     idestSkip,
      "buptim",    0,       fFalse,       kwdDest,     idestSkip,
      "colortbl", 0,        fFalse,       kwdDest,     idestSkip,
      "comment",   0,       fFalse,       kwdDest,     idestSkip,
      "creatim",   0,       fFalse,       kwdDest,     idestSkip,
      "doccomm",   0,       fFalse,       kwdDest,     idestSkip,
      "fonttbl",   0,       fFalse,       kwdDest,     idestSkip,
      "footer",    0,       fFalse,       kwdDest,     idestSkip,
      "footerf",   0,       fFalse,       kwdDest,     idestSkip,
      "footerl",   0,       fFalse,       kwdDest,     idestSkip,
      "footerr",   0,       fFalse,       kwdDest,     idestSkip,
      "footnote", 0,        fFalse,       kwdDest,     idestSkip,
      "ftncn",      0,       fFalse,       kwdDest,     idestSkip,
      "ftnsep",    0,       fFalse,       kwdDest,     idestSkip,
      "ftnsepc",   0,       fFalse,       kwdDest,     idestSkip,
      "header",    0,       fFalse,       kwdDest,     idestSkip,
      "headerf",   0,       fFalse,       kwdDest,     idestSkip,
      "headerl",   0,       fFalse,       kwdDest,     idestSkip,
      "headerr",   0,       fFalse,       kwdDest,     idestSkip,
      "info",       0,       fFalse,       kwdDest,     idestSkip,
      "keywords", 0,        fFalse,       kwdDest,     idestSkip,
      "operator", 0,        fFalse,       kwdDest,     idestSkip,
      "pict",       0,       fFalse,       kwdDest,     idestSkip,
      "printim",   0,       fFalse,       kwdDest,     idestSkip,
      "private1", 0,        fFalse,       kwdDest,     idestSkip,
      "revtim",    0,       fFalse,       kwdDest,     idestSkip,
      "rxe",        0,       fFalse,       kwdDest,     idestSkip,
      "stylesheet",    0,       fFalse,       kwdDest,     idestSkip,
      "subject",   0,       fFalse,       kwdDest,     idestSkip,
      "tc",         0,       fFalse,       kwdDest,     idestSkip,
      "title",     0,       fFalse,       kwdDest,     idestSkip,
      "txe",        0,       fFalse,       kwdDest,     idestSkip,
      "xe",         0,       fFalse,       kwdDest,     idestSkip,
      "{",          0,       fFalse,       kwdChar,     '{',
      "}",          0,       fFalse,       kwdChar,     '}',
      "\\",         0,       fFalse,       kwdChar,     '\\'
      };
int isymMax = sizeof(rgsymRtf) / sizeof(SYM);

//
// %%Function: ecApplyPropChange
//
// Set the property identified by _iprop_ to the value _val_.
//
//

int
ecApplyPropChange(IPROP iprop, int val)
{
      char *pb;

      if (rds == rdsSkip)                     // If we're skipping text,
            return ecOK;                        // don't do anything.

      switch (rgprop[iprop].prop)
      {
      case propDop:
            pb = (char *)&dop;
            break;
      case propSep:
            pb = (char *)&sep;
```

```
            break;
      case propPap:
            pb = (char *)&pap;
            break;
      case propChp:
            pb = (char *)&chp;
            break;
      default:
            if (rgprop[iprop].actn != actnSpec)
                  return ecBadTable;
            break;
      }
      switch (rgprop[iprop].actn)
      {
      case actnByte:
            pb[rgprop[iprop].offset] = (unsigned char) val;
            break;
      case actnWord:
            (*(int *) (pb+rgprop[iprop].offset)) = val;
            break;
      case actnSpec:
            return ecParseSpecialProperty(iprop, val);
            break;
      default:
            return ecBadTable;
      }
      return ecOK;
}

//
// %%Function: ecParseSpecialProperty
//
// Set a property that requires code to evaluate.
//

int
ecParseSpecialProperty(IPROP iprop, int val)
{
      switch (iprop)
      {
      case ipropPard:
            memset(&pap, 0, sizeof(pap));
            return ecOK;
      case ipropPlain:
            memset(&chp, 0, sizeof(chp));
            return ecOK;
      case ipropSectd:
            memset(&sep, 0, sizeof(sep));
            return ecOK;
      default:
            return ecBadTable;
      }
      return ecBadTable;
}

//
// %%Function: ecTranslateKeyword.
//
// Step 3.
// Search rgsymRtf for szKeyword and evaluate it appropriately.
//
// Inputs:
// szKeyword:    The RTF control to evaluate.
// param:         The parameter of the RTF control.
// fParam:        fTrue if the control had a parameter; (that is, if param is valid)
//                  fFalse if it did not.
//

int
ecTranslateKeyword(char *szKeyword, int param, bool fParam)
{
      int isym;
```

```
      // search for szKeyword in rgsymRtf

      for (isym = 0; isym < isymMax; isym++)
           if (strcmp(szKeyword, rgsymRtf[isym].szKeyword) == 0)
                break;
      if (isym == isymMax)               // control word not found
      {
           if (fSkipDestIfUnk)           // if this is a new destination
                rds = rdsSkip;             // skip the destination
                                                // else just discard it
           fSkipDestIfUnk = fFalse;
           return ecOK;
      }

      // found it!   use kwd and idx to determine what to do with it.

      fSkipDestIfUnk = fFalse;
      switch (rgsymRtf[isym].kwd)
      {
      case kwdProp:
           if (rgsymRtf[isym].fPassDflt || !fParam)
                param = rgsymRtf[isym].dflt;
           return ecApplyPropChange(rgsymRtf[isym].idx, param);
      case kwdChar:
           return ecParseChar(rgsymRtf[isym].idx);
      case kwdDest:
           return ecChangeDest(rgsymRtf[isym].idx);
      case kwdSpec:
           return ecParseSpecialKeyword(rgsymRtf[isym].idx);
      default:
           return ecBadTable;
      }
      return ecBadTable;
}

//
// %%Function: ecChangeDest
//
// Change to the destination specified by idest.
// There's usually more to do here than this...
//

int
ecChangeDest(IDEST idest)
{
      if (rds == rdsSkip)                // if we're skipping text,
           return ecOK;                    // don't do anything

      switch (idest)
      {
      default:
           rds = rdsSkip;                  // when in doubt, skip it...
           break;
      }
      return ecOK;
}

//
// %%Function: ecEndGroupAction
//
// The destination specified by rds is coming to a close.
// If there's any cleanup that needs to be done, do it now.
//

int
ecEndGroupAction(RDS rds)
{
      return ecOK;
}

//
// %%Function: ecParseSpecialKeyword
//
```

```
// Evaluate an RTF control that needs special processing.
//

int
ecParseSpecialKeyword(IPFN ipfn)
{
     if (rds == rdsSkip && ipfn != ipfnBin)   // if we're skipping, and it's not
          return ecOK;                                // the \bin keyword, ignore it.
     switch (ipfn)
     {
     case ipfnBin:
          ris = risBin;
          cbBin = lParam;
          break;
     case ipfnSkipDest:
          fSkipDestIfUnk = fTrue;
          break;
     case ipfnHex:
 ris = risHex;
 break;
     default:
          return ecBadTable;
     }
     return ecOK;
}
```

MAKEFILE

```
CFLAGS = /Zi /Od

rtfreadr.exe: rtfactn.obj rtfreadr.obj
link rtfreadr.obj rtfactn.obj,rtfreadr,rtfreadr,slibce/COD;

rtfactn.obj: rtfactn.c rtfdecl.h rtftype.h

rtfreadr.obj: rtfreadr.c rtfdecl.h rtftype.h
```

APPENDIX B: WORD 6J RICH TEXT FORMAT ADDENDUM

This document contains the changes to the Rich Text Format (RTF) specification for the Japanese version of Word 6.0 (all platforms). In this document, *Word 6J* refers to the Japanese version of Word 6.0 and *RTF-J* refers to the RTF specification described below. This document also contains some information about the interpretation of RTF-J and some behaviors of Word 6J.

This document is meant to be used in conjunction with the full RTF specification, assumes you have read that document, and does not contain the necessary information to implement an RTF reader or writer by itself. If you have any questions, please refer to the main specification first.

Appendix B Table of Contents

RTF-J

There is a Japanese local RTF specification, called RTF-J, that is somewhat different from the standard RTF specification. Although Word 6J does not write RTF-J, it can read RTF-J files. It retains the text strings in the file and disregards unknown keywords.

ESCAPED EXPRESSIONS

An escape expression (for example, \'hh, \\, or \{) is usable in all RTF control words.

Writer:

In general RTF should be written out with all characters above 0x80 in the escaped form, \'hh.

Character code	Write out as
0x00 <= ch < 0x20	Escaped (\'hh)
0x20 <= ch < 0x80	Raw (non-escaped) character
0x80 <= ch <= 0xFF	Escaped (\'hh)

For compatibility there is an RTFParam option in the [Microsoft Word] section of the WORD6.INI file (which is the same as the WINWORD6.INI file in the U.S. version) that determines whether raw 8-bit characters or escaped characters are used for the double-byte characters in **\stylesheet**, **\fonttbl**, **\bkmkstart**, and **\bkmkend**. This option is valid only when writing out the RTF; it does not affect RTF reading behavior.

[Microsoft Word]

RTFParam=7 (the default) uses an escaped expression when the character is above 0x80.

RTFParam=8 uses raw 8-bit characters for **\stylesheet**, **\fonttbl**, **\bkmkstart**, and **\bkmkend** (does not escape even if trailing-byte was an RTF special character such as \, {, or }).

Reader:

When the RTF reader encounters raw characters in the leading-byte range of the double-byte character, it regards the next character as the trailing byte of the double-byte character and combines the two characters into one double-byte character.

Leading byte	Trailing byte	Validity
Escaped	Raw (0x20 <= ch <= 0x7f)	Valid (standard format for double-byte character)
Escaped	Escaped (other)	Valid (standard format for double-byte character)
Raw	Raw	Valid (RTF-J format for double-byte character)
Raw	Escaped	Invalid

CHARACTER SET

Word 6J specifies the character set in the font table using **\fcharset**. Word 6.0J interprets **\cpg437** as **\fcharset0** and **\cpg932** as **\fcharset128** if it encounters these keywords when reading RTF. If both **\fcharset** and **\cpg** appear in the font table, **\cpg** is ignored.

Character Mapping

Word maps single-byte characters according to character set information (for example, Macintosh to ANSI) and leaves double-byte characters unmapped.

Font Family

RTF-J keywords	Definition and Word's interpretation
\jis	**RTF-J uses \jis as a keyword for character set. Word 6J interprets this as \ansi, which is the default character set used if the character set is not defined.**
\fjminchou and \fjgothic	**RTF-J uses \fjminchou and \fjgothic to specify font family. Word 6J interprets these as \fnil, which is the default font family.**

ShiftJIS Font Without \cpg or \fcharset

If **\cpg** or **\fcharset** keywords are not present, Word 6J uses the text metrics of the font before determining the character set of these fonts. If the font is unknown, Word 6J assumes it is SHIFTJIS_CHARSET.

Composite Fonts (Associated Fonts for International Runs)

Word 6J defines control words to specify composite fonts as associated character properties. These control words follow the rule of associated character properties and understand font designation (***af***). All other <aprops> are ignored in Word 6J.

<atext>	<losbrun> | <hisbrun> | <dbrun>
<losbrun>	**\hich *af** & <aprops> **\dbch *af** & <aprops> **\loch** <ptext>
<hisbrun>	**\loch *af** & <aprops> **\dbch *af** & <aprops> **\hich** <ptext>
<dbrun>	**\loch *af** & <aprops> **\hich *af** & <aprops> **\dbch** <ptext>

Control word	Definition
\loch	**Specifies a run of the characters in the low-ANSI (0x00–0x7F) area.**
\hich	For the characters in the high-ANSI (0x80—0xFF) area.
\dbch	**Specifies a run of the double-byte characters.**

Word 6J writes out associated character properties in the styles. In the style sheet, the <dbrun> definition should be used for compatibility with applications that have transparent readers.

```
{\stylesheet{\loch\af5\hich\af5\dbch\f27\fs20\snext0 Normal;}}
```

If the composite font definition matches the style, only the control word (**\loch**, **\hich**, or **\dbch**) will be used to distinguish the type of run, along with the font information for transparent readers.

```
{\fonttbl{\f5\fswiss\fcharset0\fprq2 Arial;}{\f27\froman\fcharset128\fprq1 Mincho;}}
{\stylesheet{\loch\af5\hich\af5\dbch\f27\fs20\snext0 Normal;}}
\pard\plain
{\dbch\f27\fs20 \'82\'b1\'82\'ea\'82\'cd}
{\loch\f5 Test }
{\dbch\f27\'82\'c5\'82\'b7\'81B}
\par}
```

If one or all of **\loch**, **\hich**, and **\dbch** are missing from the style sheet definition (or the character set doesn't match), Word 6J will apply appropriate fonts to each character run in the style using the bulleted rules below.

Control word	Font that Word 6J will apply
\loch	**Same font as \f.**
\hich	Any font whose character set is ANSI_CHARSET.
\dbch	**Any font whose character set is SHIFTJIS_CHARSET.**

If the composite font control words are missing from the character run, Word 6J will interpret all characters below 0x80 as a **\loch** run. Characters above or equal to 0x80 will be determined using the following rules:

- If the character is in the leading-byte range and the next character is in the trailing-byte range of a double-byte character, it will be treated as a **\dbch** run (one double-byte character). For example:

  \'99\'47→億

- If the character is in the leading-byte range of a double-byte character but the next character is not in the trailing-byte range, it will be treated as a **\hich** run (two high-ANSI or low-ANSI characters). For example:

  \'99\'FF→™ÿ

- If the character is in the leading-byte range of a double-byte character and is the last character in the run, it will be treated as a **\hich** run (one high-ANSI character). For example:

  \'99\par→™

- If the character is not in the leading-byte range of a double-byte character, it will be treated as a **\hich** run (one high-ANSI character). For example:

  \'FF→ÿ

New Control Words Created by Word 6J

ASSOCIATED CHARACTER PROPERTIES

Control word	Description
\loch	**The text consists of single-byte low-ANSI (0x00–0x7F) characters.**
\hich	The text consists of single-byte high-ANSI (0x80–0xFF) characters.
\dbch	The text consists of double-byte characters.
Borders	
\brdrdash	**Dashed border.**
\brdrdashd	Dash-dotted border.
\brdrdashdd	Dash-dot-dotted border.
Character Properties	
\uldash	**Dashed underline.**
\uldashd	Dash-dotted underline.
\uldashdd	Dash-dot-dotted underline.
\ulhair	Hairline underline.
\ulth	Thick underline.
\ulwave	Wave underline.
Document Formatting Properties	
\horzdoc	**Horizontal rendering.**
\vertdoc	Vertical rendering.
***\fchars**	List of following kinsoku characters.
***\lchars**	List of leading kinsoku characters.
\jcompress	Compressing justification (default).
\jexpand	Expanding justification.
\gutterprl	Parallel gutter.
\dgsnap	Snap to grid.
\dghspace*N*	Grid horizontal spacing in twips (the default is 120).
\dgvspace*N*	Grid vertical spacing in twips (the default is 120).
\dghorigin*N*	Grid horizontal origin in twips (the default is 1701).
\dgvorigin*N*	Grid vertical origin in twips (the default is 1984).
\dghshow*N*	Show *N*th horizontal grid (the default is 3).
\dgvshow*N*	Show ***N***th vertical grid (the default is 0).

Bullets and Numbering

\pndecd	**Double-byte decimal numbering (*arabic*dbchar).**
\pndbnum	Kanji numbering without the digit character (*dbnum1).
\pnaiu	46 phonetic katakana characters in "aiueo" order (*aiueo).
\pnaiud	46 phonetic double-byte katakana characters (*aiueo*dbchar).
\pniroha	46 phonetic katakana characters in "iroha" order (*iroha).
\pnirohad	46 phonetic double-byte katakana characters (*iroha*dbchar).
\pncnum	20 numbered list in circle (*circlenum).
\pnuldash	Dashed underline.
\pnuldashd	Dash-dotted underline.
\pnuldashdd	Dash-dot-dotted underline.
\pnulhair	Hairline underline.
\pnulth	Thick underline.
\pnulwave	Wave underline.

Drawing Objects

\dptxlrtb	**Text box flows from left to right and top to bottom (default).**
\dptxtbrl	Text box flows from right to left and top to bottom.
\dptxbtlr	Text box flows from left to right and bottom to top.
\dptxlrtbv	Text box flows from left to right and top to bottom, vertically.
\dptxtbrlv	Text box flows from top to bottom and right to left, vertically.

Index Entries

***\pxe**	**"Yomi" (pronunciation) for index entry.**

Paragraph Properties

\nocwrap	**No character wrapping.**
\nowwrap	No word wrapping.
\qd	Distributed.
\nooverflow	No overflow period and comma.
\aspalpha	Auto spacing between DBC and English.
\aspnum	Auto spacing between DBC and numbers.
\fahang	Font alignment → Hanging.
\facenter	Font alignment → Center.
\faroman	Font alignment → Roman (default).

\favar	Font alignment → Upholding variable.
\fafixed	Font alignment → Upholding fixed.

Section Formatting Properties

\horzsect	**Horizontal rendering.**
\vertsect	Vertical rendering.
\pgndecd	Double-byte decimal numbering.
\pgndbnum	Kanji numbering without the digit character.
\pgndbnumd	Kanji numbering with the digit character.

Special Characters

\zwbo	**Zero-width break opportunity. Used to insert break opportunity between two characters.**
\zwnbo	Zero-width nonbreak opportunity. Used to remove break opportunity between two characters.
\qmspace	One-quarter em space.

Tabs

\tlmdot	**Leader middle dots.**

APPENDIX C: INDEX OF RTF CONTROL WORDS

The following table contains a list of each RTF control word, the name of the section where it may be found, and a brief description of the type of control word. The types are described in the following table.

Type	Description
Flag	The control word ignores any parameter.
Destination	This control word starts a group or destination. It ignores any parameter.
Symbol	This control word represents a special character.
Toggle	This control word distinguishes between the ON and OFF states for the given property. The control word with no parameter or a nonzero parameter is used to turn on the property, while the control word with a zero parameter is used to turn it off.
Value	This control word requires a parameter.

Note: *In the following comprehensive table, the names of all control words that are new to Microsoft Word version 6.0 are followed by an asterisk (*).*

Control word	Described in section	Type
\'	Special Characters	Symbol
*****	Special Characters	Symbol
\-	Special Characters	Symbol
\:	Special Characters	Symbol
****	Special Characters	Symbol
_	Special Characters	Symbol
\{	Special Characters	Symbol
|	Special Characters	Symbol
\}	Special Characters	Symbol
\~	Special Characters	Symbol
\ab		Toggle
\absh	Positioned Objects and Frames	Value
\absw	Positioned Objects and Frames	Value
\acaps		Toggle
\acf		Value
\additive *	Style Sheet	Flag
\adn		Value
\aenddoc *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\aendnotes *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\aexpnd		Value
\af		Value
\afs		Value
\aftnbj *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\aftncn *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Destination
\aftnnalc *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\aftnnar *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag

\aftnnauc *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\aftnnchi *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\aftnnrlc *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\aftnnruc *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\aftnrestart *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\aftnrstcont *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\aftnsep *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Destination
\aftnsepc *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Destination
\aftnstart *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\aftntj *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\ai		Toggle
\alang		Value
\allprot *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\alt	Style Sheet	Flag
\annotation	Annotations	Destination
\annotprot *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\ansi	Character Set	Flag
\aoutl		Toggle
\ascaps		Toggle
\ashad		Toggle
\astrike		Toggle
\atnauthor *	Annotations	Destination
\atnicn	Annotations	Destination
\atnid	Annotations	Destination
\atnref *	Annotations	Destination
\atntime	Annotations	Destination
\atrfend *	Annotations	Destination
\atrfstart *	Annotations	Destination
\aul		Toggle
\auld		Toggle
\auldb		Toggle

\aulnone		Toggle
\aulw		Toggle
\aup		Value
\author	Information Group	Destination
\b	Character Formatting Properties	Toggle
\bgbdiag	Paragraph Shading	Flag
\bgcross	Paragraph Shading	Flag
\bgdcross	Paragraph Shading	Flag
\bgdkbdiag	Paragraph Shading	Flag
\bgdkcross	Paragraph Shading	Flag
\bgdkdcross	Paragraph Shading	Flag
\bgdkfdiag	Paragraph Shading	Flag
\bgdkhoriz	Paragraph Shading	Flag
\bgdkvert	Paragraph Shading	Flag
\bgfdiag	Paragraph Shading	Flag
\bghoriz	Paragraph Shading	Flag
\bgvert	Paragraph Shading	Flag
\bin	Pictures	Value
\binfsxn	Section Formatting Properties	Value
\binsxn	Section Formatting Properties	Value
\bkmkcolf	Bookmarks	Value
\bkmkcoll	Bookmarks	Value
\bkmkend	Bookmarks	Destination
\bkmkpub	Macintosh Edition Manager Publisher Objects	Flag
\bkmkstart	Bookmarks	Destination
\blue	Color Table	Value
\box	Paragraph Borders	Flag
\brdrb	Paragraph Borders	Flag
\brdrbar	Paragraph Borders	Flag
\brdrbtw	Paragraph Borders	Flag
\brdrcf	Paragraph Borders	Value
\brdrdash *	Paragraph Borders	Flag
\brdrdb	Paragraph Borders	Flag
\brdrdot	Paragraph Borders	Flag
\brdrhair	Paragraph Borders	Flag
\brdrl	Paragraph Borders	Flag
\brdrr	Paragraph Borders	Flag
\brdrs	Paragraph Borders	Flag
\brdrsh	Paragraph Borders	Flag
\brdrt	Paragraph Borders	Flag
\brdrth	Paragraph Borders	Flag
\brdrw	Paragraph Borders	Value
\brkfrm *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\brsp	Paragraph Borders	Value
\bullet	Special Characters	Symbol
\buptim	Information Group	Destination
\bxe	If the specified field is a form field, the *\datafield destination appears as a part of <char> and contains the binary data of a form field instruction. For example:	Flag

\caps	Character Formatting Properties	Toggle
\cb	Character Formatting Properties	Value
\cbpat	Paragraph Shading	Value
\cchs *	Character Formatting Properties	Value
\cell	Special Characters	Symbol
\cellx	Table Definitions	Value
\cf	Character Formatting Properties	Value
\cfpat	Paragraph Shading	Value
\chatn	Special Characters	Symbol
\chdate	Special Characters	Symbol
\chdpa	Special Characters	Symbol
\chdpl	Special Characters	Symbol
\chftn	Special Characters	Symbol
\chftnsep	Special Characters	Symbol
\chftnsepc	Special Characters	Symbol
\chpgn	Special Characters	Symbol
\chtime	Special Characters	Symbol
\clbgbdiag	Table Definitions	Flag
\clbgcross	Table Definitions	Flag
\clbgdcross	Table Definitions	Flag
\clbgdkbdiag	Table Definitions	Flag
\clbgdkcross	Table Definitions	Flag
\clbgdkdcross	Table Definitions	Flag
\clbgdkfdiag	Table Definitions	Flag
\clbgdkhor	Table Definitions	Flag
\clbgdkvert	Table Definitions	Flag
\clbgfdiag	Table Definitions	Flag
\clbghoriz	Table Definitions	Flag
\clbgvert	Table Definitions	Flag
\clbrdrb	Table Definitions	Flag
\clbrdrl	Table Definitions	Flag
\clbrdrr	Table Definitions	Flag
\clbrdrt	Table Definitions	Flag
\clcbpat	Table Definitions	Value
\clcfpat	Table Definitions	Value
\clmgf	Table Definitions	Flag
\clmrg	Table Definitions	Flag
\clshdng	Table Definitions	Value
\colno *	Section Formatting Properties	Value
\colortbl	Color Table	Destination
\cols	Section Formatting Properties	Value
\colsr *	Section Formatting Properties	Value
\colsx	Section Formatting Properties	Value
\column	Special Characters	Symbol
\colw *	Section Formatting Properties	Value
\comment	Information Group	Destination
\cpg	Error: Reference source not found	Value
\creatim	Information Group	Destination
\cs	Character Formatting Properties	Value
\ctrl	Style Sheet	Flag
\cvmme *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag

\datafield *	Fields	Destination
\deff	Font Table	Value
\defformat	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\deflang	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\deftab	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\deleted	Character Formatting Properties	Toggle
\dfrmtxtx	Positioned Objects and Frames	Value
\dfrmtxty	Positioned Objects and Frames	Value
\dibitmap	Pictures	Value
\dn	Character Formatting Properties	Value
\do *	Error: Reference source not found	Destination
\dobxcolumn *	Error: Reference source not found	Flag
\dobxmargin *	Error: Reference source not found	Flag
\dobxpage *	Error: Reference source not found	Flag
\dobymargin *	Error: Reference source not found	Flag
\dobypage *	Error: Reference source not found	Flag
\dobypara *	Error: Reference source not found	Flag
\doccomm	Information Group	Destination
\doctemp	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\dodhgt *	Error: Reference source not found	Value
\dolock *	Error: Reference source not found	Flag
\dpaendhol *	Error: Reference source not found	Flag
\dpaendl *	Error: Reference source not found	Value
\dpaendsol *	Error: Reference source not found	Flag
\dpaendw *	Error: Reference source not found	Value
\dparc *	Error: Reference source not found	Flag
\dparcflipx *	Error: Reference source not found	Flag
\dparcflipy *	Error: Reference source not found	Flag
\dpastarthol *	Error: Reference source not found	Flag

\dpastartl *	Error: Reference source not found	Value
\dpastartsol *	Error: Reference source not found	Flag
\dpastartw *	Error: Reference source not found	Value
\dpcallout *	Error: Reference source not found	Flag
\dpcoa *	Error: Reference source not found	Value
\dpcoaccent *	Error: Reference source not found	Flag
\dpcobestfit *	Error: Reference source not found	Flag
\dpcoborder *	Error: Reference source not found	Flag
\dpcodabs *	Error: Reference source not found	Value
\dpcodbottom *	Error: Reference source not found	Flag
\dpcodcenter *	Error: Reference source not found	Flag
\dpcodtop *	Error: Reference source not found	Flag
\dpcolength *	Error: Reference source not found	Value
\dpcominusx *	Error: Reference source not found	Flag
\dpcominusy *	Error: Reference source not found	Flag
\dpcooffset *	Error: Reference source not found	Value
\dpcosmarta *	Error: Reference source not found	Flag
\dpcotdouble *	Error: Reference source not found	Flag
\dpcotright *	Error: Reference source not found	Flag
\dpcotsingle *	Error: Reference source not found	Flag
\dpcottriple *	Error: Reference source not found	Flag
\dpcount *	Error: Reference source not found	Value
\dpellipse *	Error: Reference source not found	Flag
\dpendgroup *	Error: Reference source not found	Flag
\dpfillbgcb *	Error: Reference source not found	Value
\dpfillbgcg *	Error: Reference source not found	Value
\dpfillbgcr *	Error: Reference source not found	Value
\dpfillbggray *	Error: Reference source not found	Value

\dpfillbgpal *	Error: Reference source not found	Flag
\dpfillfgcb *	Error: Reference source not found	Value
\dpfillfgcg *	Error: Reference source not found	Value
\dpfillfgcr *	Error: Reference source not found	Value
\dpfillfggray *	Error: Reference source not found	Value
\dpfillfgpal *	Error: Reference source not found	Flag
\dpfillpat *	Error: Reference source not found	Value
\dpgroup *	Error: Reference source not found	Flag
\dpline *	Error: Reference source not found	Flag
\dplinecob *	Error: Reference source not found	Value
\dplinecog *	Error: Reference source not found	Value
\dplinecor *	Error: Reference source not found	Value
\dplinedado *	Error: Reference source not found	Flag
\dplinedadodo *	Error: Reference source not found	Flag
\dplinedash *	Error: Reference source not found	Flag
\dplinedot *	Error: Reference source not found	Flag
\dplinegray *	Error: Reference source not found	Value
\dplinehollow *	Error: Reference source not found	Flag
\dplinepal *	Error: Reference source not found	Flag
\dplinesolid *	Error: Reference source not found	Flag
\dplinew *	Error: Reference source not found	Value
\dppolycount *	Error: Reference source not found	Value
\dppolygon *	Error: Reference source not found	Flag
\dppolyline *	Error: Reference source not found	Flag
\dpptx *	Error: Reference source not found	Value
\dppty *	Error: Reference source not found	Value
\dprect *	Error: Reference source not found	Flag
\dproundr *	Error: Reference source not found	Flag

\dpshadow *	Error: Reference source not found	Flag0
\dpshadx *	Error: Reference source not found	Value
\dpshady *	Error: Reference source not found	Value
\dptxbx *	Error: Reference source not found	Flag
\dptxbxmar *	Error: Reference source not found	Value
\dptxbxtext *	Error: Reference source not found	Destination
\dpx *	Error: Reference source not found	Value
\dpxsize *	Error: Reference source not found	Value
\dpy *	Error: Reference source not found	Value
\dpysize *	Error: Reference source not found	Value
\dropcapli *	Positioned Objects and Frames	Value
\dropcapt *	Positioned Objects and Frames	Value
\ds	Section Formatting Properties	Value
\dxfrtext	Positioned Objects and Frames	Value
\dy	Information Group	Value
\edmins	Information Group	Value
\emdash	Special Characters	Symbol
\emspace *	Special Characters	Symbol
\endash	Special Characters	Symbol
\enddoc	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\endnhere	Section Formatting Properties	Flag
\endnotes	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\enspace *	Special Characters	Symbol
\expnd	Character Formatting Properties	Value
\expndtw *	Character Formatting Properties	Value
\f	Character Formatting Properties	Value
\facingp	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\falt *	Font Table	Destination
\fbidi	Font Table	Flag
\fcharset *	Font Table	Value
\fdecor	Font Table	Flag
\fet *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\fi	Paragraph Formatting Properties	Value
\fid *		Value
\field	Fields	Destination
\file *		Destination
\filetbl *		Destination

\fldalt *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\flddirty	Fields	Flag
\fldedit	Fields	Flag
\fldinst	Fields	Destination
\fldlock	Fields	Flag
\fldpriv	Fields	Flag
\fldrslt	Fields	Destination
\fmodern	Font Table	Flag
\fn	Style Sheet	Value
\fnetwork *		Flag
\fnil	Font Table	Flag
\fontemb	Font Table	Destination
\fontfile	Font Table	Destination
\fonttbl	Font Table	Destination
\footer	Headers and Footers	Destination
\footerf	Headers and Footers	Destination
\footerl	Headers and Footers	Destination
\footerr	Headers and Footers	Destination
\footery	Section Formatting Properties	Value
\footnote	Footnotes	Destination
\formdisp *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\formprot *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\formshade *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\fosnum *		Value
\fprq *	Font Table	Value
\fracwidth	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\frelative *		Value
\froman	Font Table	Flag
\fs	Character Formatting Properties	Value
\fscript	Font Table	Flag
\fswiss	Font Table	Flag
\ftech	Font Table	Flag
\ftnalt *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\ftnbj	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\ftncn	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Destination
\ftnil	Font Table	Flag
\ftnnalc *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag

\ftnnar *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\ftnnauc *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\ftnnchi *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\ftnnrlc *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\ftnnruc *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\ftnrestart	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\ftnrstcont *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\ftnrstpg *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\ftnsep	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Destination
\ftnsepc	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Destination
\ftnstart	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\ftntj	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\fttruetype	Font Table	Flag
\fvaliddos *		Flag
\fvalidhpfs *		Flag
\fvalidmac *		Flag
\fvalidntfs *		Flag
\green	Color Table	Value
\gutter	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\guttersxn	Section Formatting Properties	Value
\header	Headers and Footers	Destination
\headerf	Headers and Footers	Destination
\headerl	Headers and Footers	Destination
\headerr	Headers and Footers	Destination
\headery	Section Formatting Properties	Value
\hr	Information Group	Value
\hyphauto *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Toggle

\hyphcaps *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Toggle
\hyphconsec *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\hyphhotz	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\hyphpar *	Paragraph Formatting Properties	Toggle
\i	Character Formatting Properties	Toggle
\id	Information Group	Value
\info	Information Group	Destination
\intbl	Paragraph Formatting Properties	Flag
\ixe	If the specified field is a form field, the *\datafield destination appears as a part of <char> and contains the binary data of a form field instruction. For example:	Flag
\keep	Paragraph Formatting Properties	Flag
\keepn	Paragraph Formatting Properties	Flag
\kerning *	Character Formatting Properties	Value
\keycode	Style Sheet	Destination
\keywords	Information Group	Destination
\landscape	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\lang	Character Formatting Properties	Value
\ldblquote	Special Characters	Symbol
\level	Paragraph Formatting Properties	Value
\li	Paragraph Formatting Properties	Value
\line	Special Characters	Symbol
\linebetcol	Section Formatting Properties	Flag
\linecont	Section Formatting Properties	Flag
\linemod	Section Formatting Properties	Value
\lineppage	Section Formatting Properties	Flag
\linerestart	Section Formatting Properties	Flag
\linestart	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\linestarts	Section Formatting Properties	Value
\linex	Section Formatting Properties	Value
\linkself	Objects	Flag
\linkstyles *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\lndscpsxn	Section Formatting Properties	Flag
\lquote	Special Characters	Symbol
\ltrch	Character Formatting Properties	Flag
\ltrdoc	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\ltrmark	Special Characters	Symbol
\ltrpar	Paragraph Formatting Properties	Flag
\ltrrow	Table Definitions	Flag

\ltrsect	Section Formatting Properties	Flag
\mac	Character Set	Flag
\macpict	Pictures	Flag
\makebackup	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\margb	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\margbsxn	Section Formatting Properties	Value
\margl	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\marglsxn	Section Formatting Properties	Value
\margmirror	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\margr	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\margrsxn	Section Formatting Properties	Value
\margt	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\margtsxn	Section Formatting Properties	Value
\min	Information Group	Value
\mo	Information Group	Value
\nextfile	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Destination
\nocolbal *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\noextrasprl *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\nofchars	Information Group	Value
\nofpages	Information Group	Value
\nofwords	Information Group	Value
\noline	Paragraph Formatting Properties	Flag
\nosupersub *	Character Formatting Properties	Flag
\notabind *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\nowidctlpar *	Paragraph Formatting Properties	Flag
\nowrap	Positioned Objects and Frames	Flag
\objalias	Objects	Destination
\objalign	Objects	Value
\objautlink	Objects	Flag
\objclass	Objects	Destination
\objcropb	Objects	Value
\objcropl	Objects	Value
\objcropr	Objects	Value
\objcropt	Objects	Value
\objdata	Objects	Destination

\object	Objects	Destination
\objemb	Objects	Flag
\objh	Objects	Value
\objicemb	Objects	Flag
\objlink	Objects	Flag
\objlock	Objects	Flag
\objname	Objects	Destination
\objpub	Objects	Flag
\objscalex	Objects	Value
\objscaley	Objects	Value
\objsect	Objects	Destination
\objsetsize	Objects	Flag
\objsub	Objects	Flag
\objtime	Objects	Destination
\objtransy	Objects	Value
\objupdate *	Objects	Flag
\objw	Objects	Value
\operator	Information Group	Destination
\otblrul *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\outl	Character Formatting Properties	Toggle
\page	Special Characters	Symbol
\pagebb	Paragraph Formatting Properties	Flag
\paperh	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\paperw	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\par	Special Characters	Symbol
\pard	Paragraph Formatting Properties	Flag
\pc	Character Set	Flag
\pca	Character Set	Flag
\pghsxn	Section Formatting Properties	Value
\pgncont	Section Formatting Properties	Flag
\pgndec	Section Formatting Properties	Flag
\pgnhn *	Section Formatting Properties	Value
\pgnhnsc *	Section Formatting Properties	Flag
\pgnhnsh *	Section Formatting Properties	Flag
\pgnhnsm *	Section Formatting Properties	Flag
\pgnhnsn *	Section Formatting Properties	Flag
\pgnhnsp *	Section Formatting Properties	Flag
\pgnlcltr	Section Formatting Properties	Flag
\pgnlcrm	Section Formatting Properties	Flag
\pgnrestart	Section Formatting Properties	Flag
\pgnstart	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\pgnstarts	Section Formatting Properties	Value
\pgnucltr	Section Formatting Properties	Flag
\pgnucrm	Section Formatting Properties	Flag
\pgnx	Section Formatting Properties	Value
\pgny	Section Formatting Properties	Value

\pgwsxn	Section Formatting Properties	Value
\phcol	Positioned Objects and Frames	Flag
\phmrg	Positioned Objects and Frames	Flag
\phpg	Positioned Objects and Frames	Flag
\picbmp *	Pictures	Flag
\picbpp *	Pictures	Value
\piccropb	Pictures	Value
\piccropl	Pictures	Value
\piccropr	Pictures	Value
\piccropt	Pictures	Value
\pich	Pictures	Value
\pichgoal	Pictures	Value
\picscaled	Pictures	Flag
\picscalex	Pictures	Value
\picscaley	Pictures	Value
\pict	Pictures	Destination
\picw	Pictures	Value
\picwgoal	Pictures	Value
\plain	Character Formatting Properties	Flag
\pmmetafile	Pictures	Value
\pn *		Destination
\pnacross *		Flag
\pnb *		Toggle
\pncaps *		Toggle
\pncard *		Flag
\pncf *		Value
\pndec *		Flag
\pnf *		Value
\pnfs *		Value
\pnhang *		Flag
\pni *		Toggle
\pnindent *		Value
\pnlcltr *		Flag
\pnlcrm *		Flag
\pnlvl *		Value
\pnlvlblt *		Flag
\pnlvlbody *		Flag
\pnlvlcont *		Flag
\pnnumonce *		Flag
\pnord *		Flag
\pnordt *		Flag
\pnprev *		Flag
\pnqc *		Flag
\pnql *		Flag
\pnqr *		Flag
\pnrestart *		Flag
\pnscaps *		Toggle
\pnseclvl *		Destination
\pnsp *		Value
\pnstart *		Value
\pnstrike *		Toggle
\pntext *		Destination
\pntxta *		Destination
\pntxtb *		Destination

\pnucltr *		Flag
\pnucrm *		Flag
\pnul *		Toggle
\pnuld *		Flag
\pnuldb *		Flag
\pnulnone *		Flag
\pnulw *		Flag
\posnegx *	Positioned Objects and Frames	Value
\posnegy *	Positioned Objects and Frames	Value
\posx	Positioned Objects and Frames	Value
\posxc	Positioned Objects and Frames	Flag
\posxi	Positioned Objects and Frames	Flag
\posxl	Positioned Objects and Frames	Flag
\posxo	Positioned Objects and Frames	Flag
\posxr	Positioned Objects and Frames	Flag
\posy	Positioned Objects and Frames	Value
\posyb	Positioned Objects and Frames	Flag
\posyc	Positioned Objects and Frames	Flag
\posyil	Positioned Objects and Frames	Flag
\posyt	Positioned Objects and Frames	Flag
\prcolbl *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\printdata *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\printim	Information Group	Destination
\psover	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\psz *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\pubauto	Macintosh Edition Manager Publisher Objects	Flag
\pvmrg	Positioned Objects and Frames	Flag
\pvpara	Positioned Objects and Frames	Flag
\pvpg	Positioned Objects and Frames	Flag
\qc	Paragraph Formatting Properties	Flag
\qj	Paragraph Formatting Properties	Flag
\ql	Paragraph Formatting Properties	Flag
\qr	Paragraph Formatting Properties	Flag
\rdblquote	Special Characters	Symbol
\red	Color Table	Value
\result	Objects	Destination
\revauth *	Character Formatting Properties	Value
\revbar	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\revdttm *	Character Formatting Properties	Value
\revised	Character Formatting Properties	Toggle
\revisions	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag

\revprop	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Value
\revprot *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\revtbl *	Revision Marks	Destination
\revtim	Information Group	Destination
\ri	Paragraph Formatting Properties	Value
\row	Special Characters	Symbol
\rquote	Special Characters	Symbol
\rsltbmp	Objects	Flag
\rsltmerge	Objects	Flag
\rsltpict	Objects	Flag
\rsltrtf	Objects	Flag
\rslttxt	Objects	Flag
\rtf	RTF Version	Destination
\rtlch	Character Formatting Properties	Flag
\rtldoc	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\rtlmark	Bidirectional Language Support and Special Characters	Symbol
\rtlpar	Paragraph Formatting Properties	Flag
\rtlrow	Table Definitions	Flag
\rtlsect	Section Formatting Properties	Flag
\rxe	If the specified field is a form field, the *\datafield destination appears as a part of <char> and contains the binary data of a form field instruction. For example:	Destination
\s	Paragraph Formatting Properties	Value
\sa	Paragraph Formatting Properties	Value
\sb	Paragraph Formatting Properties	Value
\sbasedon	Style Sheet	Value
\sbkcol	Section Formatting Properties	Flag
\sbkeven	Section Formatting Properties	Flag
\sbknone	Section Formatting Properties	Flag
\sbkodd	Section Formatting Properties	Flag
\sbkpage	Section Formatting Properties	Flag
\sbys	Paragraph Formatting Properties	Flag
\scaps	Character Formatting Properties	Toggle
\sec	Information Group	Value
\sect	Special Characters	Symbol
\sectd	Section Formatting Properties	Flag
\sectnum	Special Characters	Symbol
\sectunlocked *	Section Formatting Properties	Flag
\shad	Character Formatting Properties	Toggle
\shading	Paragraph Shading	Value
\shift	Style Sheet	Flag
\sl	Paragraph Formatting Properties	Value
\slmult *	Paragraph Formatting Properties	Value
\snext	Style Sheet	Value
\softcol *	Special Characters	Flag

\softlheight *	Special Characters	Value
\softline *	Special Characters	Flag
\softpage *	Special Characters	Flag
\sprsspbf *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\sprstsp *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\strike	Character Formatting Properties	Toggle
\stylesheet	Style Sheet	Destination
\sub *	Character Formatting Properties	Flag
\subdocument *	Paragraph Formatting Properties	Value
\subject	Information Group	Destination
\super *	Character Formatting Properties	Flag
\swpbdr *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\tab	Special Characters	Symbol
\tb	Tabs	Value
\tc	Table of Contents Entries	Destination
\tcf	Table of Contents Entries	Value
\tcl	Table of Contents Entries	Value
\tcn *	Table of Contents Entries	Flag
\template	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Destination
\title	Information Group	Destination
\titlepg	Section Formatting Properties	Flag
\tldot	Tabs	Flag
\tleq	Tabs	Flag
\tlhyph	Tabs	Flag
\tlth	Tabs	Flag
\tlul	Tabs	Flag
\tqc	Tabs	Flag
\tqdec	Tabs	Flag
\tqr	Tabs	Flag
\transmf *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\trbrdrb *	Table Definitions	Flag
\trbrdrh *	Table Definitions	Flag
\trbrdrl *	Table Definitions	Flag
\trbrdrr *	Table Definitions	Flag
\trbrdrt *	Table Definitions	Flag
\trbrdrv *	Table Definitions	Flag
\trgaph	Table Definitions	Value
\trhdr *	Table Definitions	Flag
\trkeep *	Table Definitions	Flag
\trleft	Table Definitions	Value
\trowd	Table Definitions	Flag
\trqc	Table Definitions	Flag
\trql	Table Definitions	Flag
\trqr	Table Definitions	Flag

\trrh	Table Definitions	Value
\tx	Tabs	Value
\txe	If the specified field is a form field, the *\datafield destination appears as a part of <char> and contains the binary data of a form field instruction. For example:	Destination
\ul	Character Formatting Properties	Toggle
\uld	Character Formatting Properties	Flag
\uldb	Character Formatting Properties	Flag
\ulnone	Character Formatting Properties	Flag
\ulw	Character Formatting Properties	Flag
\up	Character Formatting Properties	Value
\v	Character Formatting Properties	Toggle
\vern	Information Group	Value
\version	Information Group	Value
\vertalb	Section Formatting Properties	Flag
\vertalc	Section Formatting Properties	Flag
\vertalj	Section Formatting Properties	Flag
\vertalt	Section Formatting Properties	Flag
\wbitmap	Pictures	Value
\wbmbitspixel	Pictures	Value
\wbmplanes	Pictures	Value
\wbmwidthbytes	Pictures	Value
\widowctrl	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\wmetafile	Pictures	Value
\wraptrsp *	{\info{\title The Panda's Thumb}{\ author Stephen J Gould}{\ keywords	Flag
\xe	If the specified field is a form field, the *\datafield destination appears as a part of <char> and contains the binary data of a form field instruction. For example:	Destination
\xef *	If the specified field is a form field, the *\datafield destination appears as a part of <char> and contains the binary data of a form field instruction. For example:	Value
\yr	Information Group	Value
\zwj	Special Characters	Symbol
\zwnj	Special Characters	Symbol