Microsoft® MS®-DOS, Windows®, Windows NT®, and Apple Macintosh Applications

Version: RTF Version 1.5

Microsoft Technical Support

Application Note

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

4/97– GC0165

Contents: 157 Pages, 1 Disk

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, Macintosh, and Power 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. RTF files created in Word 6.0 (and later) for the Macintosh and Power Macintosh have a file type of "RTF."

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 2 of this document). It is designed for use with the specification to assist those users developing their own RTF readers. The file included with this Application Note, Rtfreadr.exe, contains 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 Version 1.5 has been updated to include all new control words introduced by Microsoft Word for Windows 95 version 7.0 and Word 97 for Windows.

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.) There is no set maximum line length for an RTF file.

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 cannot be longer than 32 characters. 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 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 generally –32767 through 32767. However, Word tends to restrict the range to –31680 through 31680. Word allows values in the range -2,147,483,648 to 2,147,483,648 for a small number of keywords (specifically **\bin**, **\revdttm,** and some picture properties). An RTF parser must handle an arbitrary string of digits as a legal value for a keyword. 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, comments (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 on) 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 , 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. Page breaks cannot occur in destination text. Destination control words and their following text must be enclosed in braces. No other control words or text may appear within the destination group. 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>? <listtables>? <revtbl>?

Each of the various header tables should appear, if they exist, in the above order. Document properties can occur before and between the header tables. A property must be defined before being referenced. Specifically:

- The style sheet must occur before any style usage.
- The font table must precede any reference to a font.
- The **\deff** keyword must precede any text without an explicit reference to a font, because it specifies the font to use in such cases.

RTF Version

An entire RTF file is considered a group and must be enclosed in braces. The **rtf*N*** control word 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 1.5, 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)

Unicode RTF

Word 97 is a partially Unicode-enabled application. Text is handled using the 16-bit Unicode character encoding scheme. Expressing this text in RTF requires a new mechanism, because until this release (version 1.5), RTF has only handled 7-bit characters directly and 8-bit characters encoded as hexadecimal. The Unicode mechanism described here can be applied to any RTF destination or body text.

Control word	Meaning
ansicpg*N*	This keyword represents the ANSI code page which is used to perform the Unicode to ANSI conversion when writing RTF text. ***N*** represents the code page in decimal. This is typically set to the default ANSI code page of the run-time environment (for example **ansicpg1252** for U.S. Windows). The reader can use the same ANSI code page to convert ANSI text back to Unicode. This keyword should be emitted in the RTF header section right after the **ansi**, **mac**, **pc** or **pca** keyword.
upr	This keyword represents a destination with two embedded destinations, one represented using Unicode and the other using ANSI. This keyword operates in conjunction with the **ud** keyword to provide backward compatibility. The general syntax is as follows: `{\upr{keyword ansi_text}{\*\ud{keyword Unicode_text}}}` Notice that this keyword-destination does not use the * keyword; this forces the old RTF readers to pick up the ANSI representation and discard the Unicode one.
ud	This is a destination which is represented in Unicode. The text is represented using a mixture of ANSI translation and use of **u*N*** keywords to represent characters which do not have the exact ANSI equivalent.

\u*N*

This keyword represents a single Unicode character which has no equivalent ANSI representation based on the current ANSI code page. ***N*** represents the Unicode character value expressed as a decimal number.

This keyword is followed immediately by equivalent character(s) in ANSI representation. In this way, old readers will ignore the **\u*N*** keyword and pick up the ANSI representation properly. When this keyword is encountered, the reader should ignore the next ***N*** characters, where ***N*** corresponds to the last **\uc*N*** value encountered.

As with all RTF keywords, a keyword-terminating space may be present (before the ANSI characters) which is not counted in the characters to skip. While this is not likely to occur (or recommended), a **\bin** keyword, its argument, and the binary data that follows are considered one character for skipping purposes. If an RTF scope delimiter character (that is, an opening or closing brace) is encountered while scanning skippable data, the skippable data is considered to be ended before the delimiter. This makes it possible for a reader to perform some rudimentary error recovery. To include an RTF delimiter in skippable data, it must be represented using the appropriate control symbol (that is, escaped with a backslash,) as in plain text. Any RTF control word or symbol is considered a single character for the purposes of counting skippable characters.

An RTF writer, when it encounters a Unicode character with no corresponding ANSI character, should output **\u*N*** followed by the best ANSI representation it can manage. Also, if the Unicode character translates into an ANSI character stream with count of bytes differing from the current Unicode Character Byte Count, it should emit the **\uc*N*** keyword prior to the **\u*N*** keyword to notify the reader of the change.

RTF control words generally accept signed 16-bit numbers as arguments. For this reason, Unicode values greater than 32767 must be expressed as negative numbers.

\uc*N*

This keyword represents the number of bytes corresponding to a given **\u*N*** Unicode character. This keyword may be used at any time, and values are scoped like character properties. That is, a **\uc*N*** keyword applies only to text following the keyword, and within the same (or deeper) nested braces. On exiting the group, the previous **\uc** value is restored. The reader must keep a stack of counts seen and use the most recent one to skip the appropriate number of characters when it encounters a **\u*N*** keyword. When leaving an RTF group which specified a **\uc** value, the reader must revert to the previous value. A default of 1 should be assumed if no **\uc** keyword has been seen in the current or outer scopes.

A common practice is to emit no ANSI representation for Unicode characters within a Unicode destination context (that is, inside a **\ud** destination.). Typically, the destination will contain a **\uc0** control sequence. There is no need to reset the count on leaving the **\ud** destination as the scoping rules will ensure the previous value is restored.

Document Text

Document text should be emitted as ANSI characters. If there are Unicode characters that do not have corresponding ANSI characters, they should be output using the **\uc*N*** and **\u*N*** keywords.

For example, the text **LabΓValue** (Unicode characters 0x004c, 0x0061, 0x0062, 0x0393, 0x0056, 0x0061, 0x006c, 0x0075, 0x0065) should be represented as follows (assuming a previous **\ucl)**:

```
Lab\u915Gvalue
```

Destination Text

Destination text is defined as any text represented in an RTF destination. A good example is the bookmark name in the **\bkmkstart** destination.

Any destination containing Unicode characters should be emitted as two destinations within a **\upr** destination to ensure that old readers can read it properly and that no Unicode character encoding is lost when read with a new reader.

For example, a bookmark name **LabΓValue** (Unicode characters 0x004c, 0x0061, 0x0062, 0x0393, 0x0056, 0x0061, 0x006c, 0x0075, 0x0065) should be represented as follows:

```
{\upr{\*\bkmkstart LabGValue}{\*\ud{\*\bkmkstart Lab\u915 Value}}}
```

The first sub-destination contains only ANSI characters and is the representation that old readers will see. The second sub-destination is a ***\ud** destination which contains a second copy of the **\bkmkstart** destination. This copy can contain Unicode characters and is the representation that Unicode-aware readers must pay attention to, ignoring the ANSI-only version.

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 syntax listed in the following table.

<fonttbl>	'{' **fonttbl** (<fontinfo> | ('{' <fontinfo> '}'))+ '}'
<fontinfo>	<fontnum><fontfamily><fcharset>?<fprq>?<panose>?<nontaggedname>?<fontemb>? <codepage>? <fontname><fontaltname>? ';'
<fontnum>	**\f**
<fontfamily>	**\fnil** | **\froman** | **\fswiss** | **\fmodern** | **\fscript** | **\fdecor** | **\ftech** | **\fbidi**
<fcharset>	***\fcharset***
<fprq>	***\fprq***
<panose>	<data>
<nontaggedname>	***fname**
<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, pitch of a font in the font table, and non-tagged font name.

Control word	Definition
\fcharset*N*	Specifies the character set of a font in the font table. Values for N are defined by Windows header files, and in the file RTFDEFS.H accompanying this document.
\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.
***panose**	Destination keyword. This destination contains a 10-byte Panose 1 number. Each byte represents a single font property as described by the Panose 1 standard specification.
***fname**	This is an optional control word in the font table to define the non-tagged font name. This is the actual name of the font without the tag, used to show which character set is being used. For example, Arial is a non-tagged font name, and Arial (Cyrillic) is a tagged font name. This control word is used by WordPad. Word ignores this control word (and never creates it).
\fbias*N*	Used to arbitrate between two fonts when a particular character can exist in either non-Far East or Far East font. Word 97 emits the **\fbiasN** keyword only in the context of bullets or list information (that is, a **\listlevel** destination). The default value of 0 for **N** indicates a non-Far East font. A value of 1 indicates a Far East font. Additional values may be defined in future releases.

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 file name, or the actual font data may be located inside the group. If a file name is specified, it is contained in the **fontfile** group. The **cpg** control word can be used to specify the character set for the file name.

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, file names (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 file names as well. However, all RTF documents must still declare a character set (that is, **ansi,** **mac,** **pc,** or **pca**) to maintain backward 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. The only time a file table is created in RTF is when the document contains subdocuments. This group defines the files referenced in the document and has the following syntax:

<filetbl> '{*' **filetbl** ('{' <fileinfo> '}')+ '}'

<fileinfo>	**file** <filenum><relpath>?<osnum>? <filesource>+ <file name>
<filenum>	***\fid***
<relpath>	***frelative***
<osnum>	***fosnum***
<filesource>	**fvalidmac** | **fvaliddos** | **fvalidntfs** | **fvalidhpfs** | **fnetwork**
<file name>	#PCDATA

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

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) where the referenced file's path starts to be relative to the path of the owning document. For example, if 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. The example below defines the default color table used by Word. The first color is omitted, as shown by the semicolon following the **\colortbl** control word. The missing definition indicates that color 0 is the ''auto'' color.

```
{\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;\red0
\green0\blue128;\red0\green128\blue128;\red0\green128\blue0;\red128\green0\blue128;\red128\green
0\blue0;\red128\green128\blue0;\red128\green128\blue128;\red192\green192\blue192;}
```

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***

<autoupd> ***\sautoupd***

<hidden> ***\shidden***

<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. Like **\s**, **\cs** is not a destination control word. However, it is important to treat it like one inside the style sheet; that is, **\cs** must be prefixed with * and must appear as the first item inside a group. Doing so ensures that readers that do not understand character styles will skip the character style information correctly. When used in body text to indicate that a character style has been applied, do not include the * prefix.
\s*N*	Designates paragraph style.
\ds*N*	Designates section style.
\additive	Used in a character style definition ('**{*'\cs**_'}'). Indicates that character style attributes are to be added to the current paragraph style attributes, rather than setting the paragraph attributes to only those defined in the character 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.
\sautoupd	Automatically update styles.
\shidden	Style does not appear in the **Styles** drop-down list in the **Style** dialog box[1] (on the **Format** menu, click **Styles**).
\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
```

[1] The hidden style property can only be accessed using Visual Basic for Applications.

```
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.

List Table

Word 97 stores bullets and numbering information very differently from earlier versions of Word. In Word 6.0, for example, number formatting data is stored individually with each paragraph. In Word 97, however, all of the formatting information is stored in a pair of document-wide list tables which act as a style sheet, and each individual paragraph stores only an index to one of the tables, like a style index.

There are two list tables in Word: the List table (destination **\listtable**), and the List Override table (destination **\listoverridetable).**

The first table Word stores is the List table. A List table is a list of lists (destination **\list**). Each list contains a number of list properties that pertain to the entire list, and a list of levels (destination **\listlevel)**, each of which contains properties that pertain only to that level.

Top-level List Properties

Control word	Meaning
\listid*N*	Each list must have a unique list ID that should be randomly generated. The value ***N*** is a long integer. The list ID cannot be between -1 and -5.
\listtemplateid*N*	Each list should have a unique template ID as well, which also should be randomly generated. The template ID cannot be -1. The value ***N*** is a long integer.
\listsimple*N*	1 if the list has one level; 0 if the list has nine levels
\listrestarthdn*N*	1 if the list restarts at each section; 0 if not. Used for Word 7.0 compatibility only.
\listname	The argument for **\listname** is a string that is the name of this list. Names allow ListNum fields to specify the list they belong to. This is a destination control word.

List Levels

Each list consists of either one or nine list levels depending upon whether the **\listsimple** flag is set. Each list level contains a number of properties that specify the formatting for that level, such as the start-at value, the text string surrounding the number, its justification and indents, and so on.

Control word	Meaning
\levelstartat*N*	***N*** specifies the start-at value for the level

\levelnfc*N*	Specifies the number type for the level: 0 Arabic (1, 2, 3) 1 Uppercase Roman numeral (I, II, III) 2 Lowercase Roman numeral (i, ii, iii) 3 Uppercase letter (A, B, C) 4 Lowercase letter (a, b, c) 5 Ordinal number (1st, 2nd, 3rd) 6 Cardinal text number (One, Two Three) 7 Ordinal text number (First, Second, Third) 22 Arabic with leading zero (01, 02, 03, ..., 10, 11) 23 Bullet (no number at all) 255 No number
\leveljc*N*	0 Left justified 1 Center justified 2 Right justified
\levelold*N*	1 if this level was converted from Word 6.0 or 7.0, 0 if it is a native Word 97 level.
\levelprev*N*	1 if this level includes the text from the previous level (used for Word 7.0 compatibility only); otherwise, the value is 0 . This keyword will only be valid if the **\leveloldN** keyword is emitted.
\levelprevspace*N*	1 if this level includes the indentation from the previous level (used for Word 7.0 compatibility only); otherwise, the value is 0 . This keyword will only be valid if the **\leveloldN** keyword is emitted.
\levelindent*N*	Minimum distance from the left indent to the start of the paragraph text (used for Word 7.0 compatibility only). This keyword will only be valid if the \levelold*N* keyword is emitted.
\levelspace*N*	Minimum distance from the right edge of the number to the start of the paragraph text (used for Word 7.0 compatibility only). This keyword will only be valid if the **\levelold*N*** keyword is emitted.
\leveltext	The argument for this level should be the number format string for this level. The first character is the length of the string, and any numbers within the level should be replaced by the index of the level they represent. For example, a level three number such as “1.1.1.” would generate the following RTF: “**{\leveltext \’06\’00.\’01.\’02.}**” where the ’06 is the string length, the \’00, \’01, and \’02 are the level place holders, and the periods are the surrounding text. This is a destination control word.

\levelnumbers	The argument for this destination should be a string that gives the offsets into the **\leveltext** of the level place holders. In the above example, "1.1.1.", the **\levelnumbers** RTF should be `{\levelnumbers \'01\'03\'05}` because the level place holders have indices 1, 3, and 5. This is a destination control word.
\levelfollow*N*	Specifies which character follows the level text: 0 Tab 1 Space 2 Nothing
\levellegal*N*	1 if any list numbers from previous levels should be converted to arabic numbers; 0 if they should be left with the format specified by their own level's definition.
\levelnorestart*N*	1 if this level does not restart its count each time a number of a higher level is reached, 0 if this level does restart its count each time a number of a higher level is reached.

In addition to all of these properties, each list level can contain any character properties (all of which affect all text for that level) and any combination of three paragraph properties: left indents, first line left indents, and tabs—each of which must be of a special type: **jclisttab**. These paragraph properties will be automatically applied to any paragraph in the list.

List Override Table

The List Override table is a list of list overrides (destination **listoverride**). Each list override contains the **listid** of one of the lists in the List table, as well as a list of any properties it chooses to override. Each paragraph will contain a list override index (keyword **ls**) which is a 1-based index into this table. Most list overrides don't override any properties—instead, they provide a level of indirection to a list. There are generally two types of list overrides: (1) formatting overrides, which allow a paragraph to be part of a list and are numbered along with the other members of the list, but have different formatting properties; and, (2) start-at overrides, which allow a paragraph to share the formatting properties of a list, but have a different start-at values. The first element in the document with each list override index takes the start-at value that the list override specifies as its value, while each subsequent element is assigned the number succeeding the previous element of the list.

List overrides have a few top-level keywords, including a **listoverridecount**, which contains a count of the number of levels whose format is overridden. This **listoverridecount** should always be either 1 or 9, depending upon whether the list to be overridden is simple or multilevel. All of the actual override information is stored within a list of list override levels (destination **lfolevel**).

Control word	Meaning
\listid*N*	Should exactly match the **\listid** of one of the lists in the List table. The value ***N*** is a long integer.
\listoverridecount*N*	Number of list override levels within this list override (from 0 or 9).
\ls	The (1-based) index of this **\listoverride** in the **\listoverride** table. This value should never be zero inside a **\listoverride**, and must be unique for all **\listoverrides** within a document. The valid values are from 1 to 2000.

List Override Level

Each list override level contains flags to specify whether the formatting or start-at values are being overridden for each level. If the format flag (**listoverrideformat**) is given, the **lfolevel** should also contain a list level (**listlevel**). If the start-at flag (**listoverridestart**) is given, a start-at value must be provided. If the start-at is overridden but the format is not, then a **levelstartat** should be provided in the **lfolevel** itself. If both start-at and format are overridden, put the **levelstartat** inside the **listlevel** contained in the **lfolevel**.

Control word	Meaning
\listoverridestart*N*	Should exactly match the **listID** of one of the lists in the List table. The value ***N*** is a long integer.
\listoverrideformat*N*	Number of list override levels within this list override (should be either 1 or 9).

Track Changes (Revision Marks)

This table allows tracking of multiple authors and reviewers of a document, and is used in conjunction with the character properties for tracking changes (using 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 hexadecimal values enclosed in quotation marks.

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>? & <manager>? & <company>? <operator>? & <category>? & <keywords>? & <comment>? & ***\version***? & <doccomm>? & ***\vern***? & <creatim>? & <revtim>? & <printim>? & <buptim>? & ***\edmins***? & ***\nofpages***? & ***\nofwords***? ***\nofchars***? & ***\id***? '}'
<title>	'{' **\title** #PCDATA '}'
<subject>	'{' **\subject** #PCDATA '}'
<author>	'{' **\author** #PCDATA '}'
<manager>	{' **\manager** #PCDATA '}'
<company>	{' **\company** #PCDATA '}'
<operator>	'{' **\operator** #PCDATA '}'
<category>	{' **\category** #PCDATA '}'
<keywords>	'{' **\keywords** #PCDATA '}'
<comment>	'{' **\comment** #PCDATA '}'
<doccomm>	'{' **\doccomm** #PCDATA '}'
<hlinkbase>	'{' **\hlinkbase** #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.
\manager	Manager of the author. This is a destination control word.
\company	Company of the author. This is a destination control word
\operator	Person who last made changes to the document. This is a destination control word.
\category	Category of 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 the **Summary Info** or **Properties** dialog box in Word. This is a destination control word.
\hlinkbase	The base address that is used for the path of all relative hyperlinks inserted in the document. This can be a path or an Internet address (URL).

The **userprops** control word introduces the user-defined document properties. Unique **propname** control words define each user-defined property in the document. The group has the following syntax.

\userprops	'{*' **userprops** ('{' <propinfo> '}'*) '}'
<propinfo>	<propname><proptype><staticval><linkval>?
<propname>	'{' **propname** #PCDATA '}'
<proptype>	**proptype**
<staticval>	**staticval**
<linkval>	**linkval**

Control Word	**Meaning**
\propname	The name of the user-defined property.
\staticval	The value of the property.
\linkval	The name of a bookmark that contains the text to display as the value of the property.
\proptype	**proptypeN** Specifies the type of the property.

For **proptype,** the ***N*** argument can have the following values.

Value	**Description**
3	Integer
5	Real number
7	Date
11	Boolean
30	Text

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 including spaces
\nofcharsws	Number of characters not including spaces
\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 there are 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 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 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 34 of this Application Note for a list of possible values for ***N***.
\deflangfe	Default language ID for Asian versions of Word.
\windowcaption	Sets the caption text for the document window. This is a string value.
\doctype*N*	An integer (0-2) that describes the document type for AutoFormat. 0 General Document (for formatting most documents, the default) 1 Letter (for formatting letters, and used by Letter Wizard) 2 E-mail (for formatting e-mail, and used by WordMail)
\fromtext	Indicates document was originally plain text.

Document views and zoom level

\viewkind*N*	An integer (0-5) that represents the view mode of the document. 0 None 1 Page Layout view 2 Outline view 3 Master Document view 4 Normal view 5 Online Layout view
\viewscale*N*	Zoom level of the document; the ***N*** argument is a value representing a percentage (the default is 100).
\viewzk*N*	An integer (0 to 2) that represents the zoom kind of the document. 0 None 1 Full page 2 Best fit
\private	Obsolete destination. It has no leading *. It should be skipped.

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** readers 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.
\sprslnsp	Suppress extra line spacing like WordPerfect version 5.x.
\subfontby size	Substitute fonts based on size first.
\truncatefont height	Round down to the nearest font size instead of rounding up.
\truncex	Don't add leading (extra space) between rows of text
\bdbfhdr	Print body before header/footer. Option for compatibility with Word for the Macintosh 5.x.
\dntblnsbdb	Don't balance SBCS/DBCS characters. Option for compatibility with Word 6.0 (Japanese).
\expshrtn	Expand character spaces on line-ending with SHIFT+RETURN. Option for compatibility with Word 6.0 (Japanese).
\lytexcttp	Don’t center exact line height lines.
\lytprtmet	Use printer metrics to lay out document.
\msmcap	Small caps like Word for the Macintosh 5.x.
\nolead	No external leading. Option for compatibility with Word for the Macintosh 5.x
\nospaceforul	Don't add space for underline. Option for compatibility with Word 6.0 (Japanese).
\noultrlspc	Don't underline trailing spaces. Option for compatibility with Word 6.0 (Japanese).
\noxlattoyen	Don't translate backslash to Yen sign. Option for compatibility with Word 6.0 (Japanese).
\oldlinewrap	Lines wrap like Word 6.0.
\sprsbsp	Suppress extra line spacing at bottom of page.
\sprstsm	Does nothing. This keyword should be ignored.
\wpjst	Do full justification like WordPerfect 6.x for Windows.
\wpsp	Set the width of a space like WordPerfect 5.x.
\wptab	Advance to next tab stop like WordPerfect 6.x.
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 box 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).

Comments (Annotations)

\annotprot	This document is protected for comments (annotations). The user cannot edit the document but can insert comments (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” in this Application Note.

Page Borders

\pgbrdrhead	Page border surrounds header.
\pgbrdrfoot	Page border surrounds footer.
\pgbrdrt	Page border top.
\pgbrdrb	Page border bottom.
\pgbrdrl	Page border left.
\pgbrdrr	Page border right.
\brdrart*N*	Page border art; the ***N*** argument is a value from 1-165 representing the number of the border.
\pgbrdropt*N*	8 Page border measure from text. **Always display in fron**t option is set to **off**.
	32 Page border measure from edge of page. **Always display in front** option is set to **on**.
	40 Page border measure from edge of page. **Always display in front** option is set to **off.**
\pgbrdrsnap	Align paragraph borders and table edges with page border.

The color, width, border style, and border spacing keywords for page borders are the same as the keywords defined for paragraph borders.

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.
\margmirsxn	Switches margin definitions on left and right pages. Used in conjunction with **facingp**.
\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.
\headery*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). This control word is understood but not used by current versions (6.0 or later) of Word.
\pgny*N*	Page number is ***N*** twips from the top margin (the default is 720). This control word is understood but not used by current versions (6.0 or later) of Word.
\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).

Text Flow

\stextflow Section property for specifying text flow.

0	Text flows left to right and top to bottom
1	Text flows top to bottom and right to left, vertical
2	Text flows left to right and bottom to top
3	Text flows right to left and top to bottom
4	Text flows left to right and top to bottom, vertical
5	Text flows vertically, non-vertical font

Page Borders

\pgbrdrhead	Page border surrounds header.
\pgbrdrfoot	Page border surrounds footer.
\pgbrdrt	Page border top.
\pgbrdrb	Page border bottom.
\pgbrdrl	Page border left.
\pgbrdrr	Page border right.
\brdrart*N*	Page border art; the ***N*** argument is a value from 1-165 representing the number of the border.

\pgbrdropt*N*	8 Page border measure from text. **Always display in fron**t option is set to **off**.
	32 Page border measure from edge of page. **Always display in front** option is set to **on**.
	40 Page border measure from edge of page. **Always display in front** option is set to **off**.
\pgbrdrsnap	Align paragraph borders and table edges with page border.

The color, width, border style, and border spacing keywords for page borders are the same as the keywords defined for paragraph borders.

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 "Document Formatting Properties" on page 30 and "Section Formatting Properties" on page 30 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 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**.
widctlpar	Widow/orphan control is used for the current paragraph. This is a paragraph property used to override the absence of 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.
outlinelevel*N*	Outline level of paragraph. The ***N*** argument is a value from 0-8 representing the outline level of the paragraph. In the default case, no outline level is specified (same as body text).
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

Word 6.0/95 RTF

To provide compatibility with existing RTF readers, all applications that can automatically format paragraphs with bullets or numbers 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 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.

Word 97 RTF

Each paragraph that is part of a list must contain some keyword to indicate which list it's in, and which level of the list it belongs to. Word 97 also provides the flat text representation of each number (in the **\listtext** destination); so, RTF readers that don't understand Word 97 numbering will get the paragraph number, along with appropriate character properties, inserted into their document at the beginning of the paragraph. Any RTF reader that does understand Word 97 numbering should ignore the entire **listtext** destination.

Control word	Meaning
ls	Should exactly match the **ls** for one of the list overrides in the List Override table.
ilvl	The 0-based level of the list to which the paragraph belongs. For all simple lists, this should always be 0 . For multilevel lists, it can be 0-8.
listtext	Contains the flat text representation of the number, including character properties. Should be ignored by any reader that understands Word 97 numbering. This is a destination control word.

REVISION MARKS FOR PARAGRAPH NUMBERS

Paragraph numbers and ListNum fields track revision information with special properties applied to the paragraph mark and ListNum field, respectively. The special properties hold the "old" value of the number—the value it held when revision-mark tracking began. At display time, Word checks the number's current value and compares it with this "old" value to tell if it has changed. If the numbers are different, the old value shows up as deleted and the new value as inserted; if the numbers are the same, Word displays the new value normally, with no revision information. If there was no old value, the new value shows up as inserted. The following table lists the RTF specifications for these special properties.

Track Changes (Revision mark) properties for paragraph numbers	
pnrauth*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. **Note** This keyword is used to indicate paragraph number revisions.
pnrdate*N*	Time of the revision. The 32-bit DTTM structure is emitted as a long integer.
pnrnot	Indicates if the paragraph number for the current paragraph is marked as "inserted."
pnrxst*N*	The keywords **pnrxst**, **pnrrgb**, **pnrpnbr**, and **pnrnfc** describe the "deleted number" text for the paragraph number. Their values are binary. Each of these keywords is represented as an array. The deleted number is written out with a **pnrstart** keyword, followed by the array's keyword, followed by the first byte of the array, followed by the array's keyword, followed by the second byte of the array's keyword, followed by the array's keyword, followed by the third byte of the array's keyword, and so on. This sequence is followed by the **pnrstop** keyword. **pnrxst** is a 32-item Unicode character array (double bytes for each character) with a length byte as the first number—it has the actual text of the number, with "level" place holders written out as digits from 0-8.
pnrrgb*N*	Nine-item array of indices of the level place holders in the **pnrxst** array.
pnrnfc*N*	Nine-item array containing the number format codes of each level (using the same values as the **levelnfc** keyword). The number format code is represented as a short integer.

pnrpnbr*N*	Nine-item array of the actual values of the number in each level. The number is represented as a long integer
pnrstart*N*	The **pnrxst,** **pnrrgb,** **pnrpnbr,** and **pnrnfc** arrays are each preceded by the **pnrstart** keyword, whose argument is 0-3 depending on the array.
pnrstop*N*	The **pnrxst,** **pnrrgb,** **pnrpnbr,** and **pnrnfc** arrays are each terminated by the **pnrstop** keyword, whose argument is the number of bytes written out in the array.

Example:

Let's take an example of the number "3-4b." which represents the third level of the list. The following table lists the values of each array.

Array	Binary	Comment
pnrxst	\'05\'00-\'01\'02	The length of the string is 5. Then, first level (level 0), followed by a dash, followed by the second and third levels (levels 1 and 2), followed by a period.
Pnrrgb	\'01\'03\'04	The level place holders are at indices 1, 3, and 4 in the string.
Pnrnfc	\'00\'00\'04	The nfc values are Arabic (0), Arabic (0), and lowercase letter (4).
Pnrpnbr	\'03\'04\'02	The numbers or 3, 4, and 2 (b)

Here is the RTF for this number:

```
\pnrstart0
\pnrxst0\pnrxst5\pnrxst0\pnrxst1\pnrxst0\pnrxst45\pnrxst0\pnrxst2\pnrxst0\pnrxst3\pnrxst0\pnrxst4
6
\pnrstop12

\pnrstart1
\pnrrgb1\pnrrgb3\pnrrgb4
\pnrrgb0\pnrrgb0\pnrrgb0
\pnrrgb0\pnrrgb0\pnrrgb0
\pnrstop9

\pnrstart2
\pnrnfc0\pnrnfc0\pnrnfc0\pnrnfc0\pnrnfc0\pnrnfc4
\pnrnfc0\pnrnfc0\pnrnfc0\pnrnfc0\pnrnfc0\pnrnfc0
\pnrnfc0\pnrnfc0\pnrnfc0\pnrnfc0\pnrnfc0\pnrnfc0
\pnrstop18

\pnrstart3
\pnrpnbr0\pnrpnbr0\pnrpnbr0\pnrpnbr3
\pnrpnbr0\pnrpnbr0\pnrpnbr0\pnrpnbr4
\pnrpnbr0\pnrpnbr0\pnrpnbr0\pnrpnbr2
\pnrpnbr0\pnrpnbr0\pnrpnbr0\pnrpnbr0
\pnrpnbr0\pnrpnbr0\pnrpnbr0\pnrpnbr0
\pnrpnbr0\pnrpnbr0\pnrpnbr0\pnrpnbr0
\pnrpnbr0\pnrpnbr0\pnrpnbr0\pnrpnbr0
\pnrpnbr0\pnrpnbr0\pnrpnbr0\pnrpnbr0
```

```
\pnrpnbr0\pnrpnbr0\pnrpnbr0\pnrpnbr0
\pnrstop36
```

Control word	Meaning
Track Changes (Revision mark) properties for ListNum fields	
\dfrauth*N*	Index into the revision table. The content of the ***N***th group in the revision table is considered the author of that revision. **Note** This keyword is used to indicate the deleted value of a ListNum field.
\dfrdate*N*	Time of the revision. The 32-bit DTTM structure is emitted as a long integer.
\dfrxst	Unicode character array with a length byte.
\dfrstart	The \dfrxst array is preceded by the **\dfrstart** keyword.
\dfrstop	The \dfrxst array is terminated by the **\dfrstop** keyword.

Example:

Let's take the sample example from above. The deleted value is "3-4b." The RTF would then be

```
\dfrstart0\dfrxst0\dfrxst5\dfrxst0\dfrxst51\dfrxst0\dfrxst45\dfrxst0\dfrxst52
\dfrxst0\dfrxst66\dfrxst0\dfrxst46\dfrstop10
```

where 5 is the length byte, 51 is Unicode for "3", 45 is Unicode for "-", 52 is Unicode for "4", and so on.

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.
\brdrdashsm	Dash border (small).
\brdrdashd	Dot dash border.
\brdrdashdd	Dot dot dash border.
\brdrtriple	Triple border.
\brdrtnthsg	Thick thin border (small).
\brdrthtnsg	Thin thick border (small).
\brdrtnthtnsg	Thin thick thin border (small).
\brdrtnthmg	Thick thin border (medium).
\brdrthtnmg	Thin thick border (medium).
\brdrtnthtnmg	Thin thick thin border (medium).
\brdrtnthlg	Thick thin border (large).
\brdrthtnlg	Thin thick border (large).
\brdrtnthtnlg	Thin thick thin border (large).
\brdrwavy	Wavy border.
\brdrwavydb	Double wavy border.
\brdrdashdotstr	Striped border.
\brdremboss	Emboss border.
\brdrengrave	Engrave border.
\brdrw*N*	***N*** is the width in twips of the pen used to draw the paragraph border line. ***N*** cannot be greater than 75. To obtain a larger border width, the ***brdth*** control word can be used to obtain a width double that of ***N.***
\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 fill color, 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 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 re-specifying 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**? & **\abslock**

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.
abslock	Locks a frame anchor to the current paragraph that it is associated with.

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.
overlay	Text flows underneath frame.
posyin	Positions the paragraph vertically inside the reference frame.
posyout	Positions the paragraph vertically outside the reference 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.
tcelld	Sets table cell 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, 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 to be 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.

Vertical Text Alignment	
\clvertalt	Text is top-aligned in cell (the default).
\clvertalc	Text is centered vertically in cell.
\clvertalb	Text is bottom-aligned in cell.
\cltxlrtb	Vertical text aligned left (direction bottom up).
\cltxtbrl	Vertical text aligned right (direction top down).

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>

Font (character) Formatting Properties

These control words (described as <chrfmt> in the syntax description) change font (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 . For example, **b** turns on bold, while **b0** turns off bold.

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

Control word	Meaning
plain	Reset font (character) formatting properties to a default value defined by the application (for example, bold, underline and italic are disabled; font size is reset to 12 pt). The associated font (character) formatting properties (described in the section "Associated Font (character) Properties" on page 37 of this Application Note) are also reset.
animtext*N*	Animated text properties. 1 Las Vegas Lights 2 Blinking background 3 Sparkle text 4 Marching black ants 5 Marching red ants 6 Shimmer
b	Bold.*
caps	All capitals.*
charscalex*N*	Character scaling value. The ***N*** argument is a value representing a percentage (the default is 100).
deleted	Marks the text as deletion revision marked.*
dn*N*	Subscript position in half-points (the default is 6).
embo	Emboss.
impr	Engrave.

\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.*
\scaps	Small capitals.*
\shad	Shadow.*
\strike	Strikethrough.*
\strikedl	Double strikethrough.
\ul	Continuous underline. **\ul0** turns off all underlining.
\uld	Dotted underline.
\uldash	Dash underline.
\uldashd	Dot dash underline.
\uldashdd	Dot dot dash underline.
\uldb	Double underline.
\ulnone	Stops all underlining.
\ulth	Thick underline
\ulw	Word underline.
\ulwave	Wave 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
Sesotho (Sotho)	0x0430
Afrikaans	0x0436
Zulu	0x0435
Xhosa	0x0434
Venda	0x0433
Tswana	0x0432
Tsonga	0x0431
Farsi (Persian)	0x0429

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).

Character Borders and Shading

Character shading has the following syntax.

<shading>	(***chshdng*** | <pat>) ***chcfpat***? ***chcbpat***?
<pat>	**chbghoriz** | **chbgvert** | **chbgfdiag** | **chbgbdiag** | **chbgcross** | **chbgdcross** | **chbgdkhoriz** | **chbgdkvert** | **chbgdkfdiag** | **chbgdkbdiag** | **chbgdkcross** | **chbgdkdcross**

Control word	**Meaning**
chbrdr	Character border (border always appears on all sides).
chshdng*N*	Character shading. The ***N*** argument is a value representing the shading of the text in hundredths of a percent.

\chcfpat*N*	***N*** is the color of the background pattern, specified as an index into the document's color table.
\chcbpat*N*	***N*** is the fill color, specified as an index into the document's color table.
\chbghoriz	Specifies a horizontal background pattern for the text.
\chbgvert	Specifies a vertical background pattern for the text.
\chbgfdiag	Specifies a forward diagonal background pattern for the text (\\\\).
\chbgbdiag	Specifies a backward diagonal background pattern for the text (////).
\chbgcross	Specifies a cross background pattern for the text.
\chbgdcross	Specifies a diagonal cross background pattern for the text.
\chbgdkhoriz	Specifies a dark horizontal background pattern for the text.
\chbgdkvert	Specifies a dark vertical background pattern for the text.
\chbgdkfdiag	Specifies a dark forward diagonal background pattern for the text (\\\\).
\chbgdkbdiag	Specifies a dark backward diagonal background pattern for the text (////).
\chbgdkcross	Specifies a dark cross background pattern for the text.
\chbgdkdcross	Specifies a dark diagonal cross background pattern for the text.

The color, width, and border style keywords for character borders are the same as the keywords for paragraph borders.

Control word	**Meaning**
Track Changes (Revision Mark) properties	
\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.
\crauth*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. **Note** This keyword is used to indicate formatting revisions, such as bold, italic, and so on.
\crdate*N*	Time of the revision. The 32-bit DTTM structure is emitted as a long integer.
\revauthdel*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 deletion.
\revdttmdel*N*	Time of the deletion. The 32-bit DTTM structure is emitted as a long integer.

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. Although RTF defines a broad variety of associated character properties, any implementation may choose not to 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 .

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 described on page 35 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).

Highlighting

This property applies highlighting to text. The formatting is not a character format, so it cannot be part of a style definition.

Control Word	Definition
highlight*N*	Highlights the specified text. ***N*** specifies the color.

For **highlight**, the ***N*** argument can have the following values:

Value	Description
1	Black
2	Blue
3	Cyan
4	Green
5	Magenta
6	Red
7	Yellow
8	Unused
9	Dark Blue
10	Dark Cyan
11	Dark Green
12	Dark Magenta
13	Dark Red
14	Dark Yellow
15	Dark Gray
16	Light Gray

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, 1997).
chdpa	Current date in abbreviated format (for example, Thu, Oct 28, 1997).
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.
\emdash	Em-dash (—).
\endash	En-dash (–).
\emspace	Nonbreaking space equal to width of character "m" in current font. Some old RTF writers use the construct ‘{**emspace** }’ (with two spaces before the closing brace) to trick readers unaware of **emspace** into parsing a regular space. A reader should interpret this as an **emspace** and a regular space.
\enspace	Nonbreaking space equal to width of character "n" in current font. Some old RTF writers use the construct ‘{**enspace** }’ (with two spaces before the closing brace) to trick readers unaware of **enspace** into parsing a regular space. A correct reader should interpret this as an **enspace** and a regular space.
\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. (Used by Word 5.1 for the Macintosh as the beginning delimiter for a string of formula typesetting commands.)
\~	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 for ligating (joining) characters.
\zwnj	Zero-width nonjoiner. This is used for unligating a 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; otherwise, 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.)

A tab (character value 9) should be treated as a **tab** control word. Not all RTF readers understand this; therefore, an RTF writer should always emit the control word for tabs.

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

Document Variables

Document variables are definable and accessed through macros. The group has the following syntax.

<variables>	‘{*’ <docvar>‘{’ <varname> ‘}’ ‘{’ <vartext> ‘}’ ‘}’*
<docvar>	**\docvar**
<varname>	#PCDATA
<vartype>	#PCDATA

Control Word	Definition
\ **docvar**	A group that defines a document variable name and its value.

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 is "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. The **\pict** keyword is preceded by ***\shppict** destination control keyword as described in the following example. A picture destination has the following syntax:

<pict>	'{' **\pict** (<brdr>? & <shading>? & <picttype> & <pictsize> & <metafileinfo>?) <data> '}'
<picttype>	| **\emfblip** | **\pngblip** | **\jpegblip | \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
\emfblip	Source of the picture is an EMF (enhanced metafile).
\pngblip	Source of the picture is a PNG.
\jpegblip	Source of the picture is a JPEG.
\shppict	Specifies a Word 97 picture. This is a destination control word.
\nonshppict	Specifies that Word 97 has written a `{\pict` destination that it will not read on input. This keyword is for compatibility with other readers.

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 described on page 43 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.

Example:

```
{\*\shppict {\pict \emfblip ….. }}{\nonshppict {\pict ….}}
```

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 because 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. The ***N*** argument is a long integer.
pich*N*	*yExt* field if the picture is a Windows metafile; picture height in pixels if the picture is a bitmap or from QuickDraw. The ***N*** argument is a long integer.
picwgoal*N*	Desired width of the picture in twips. The ***N*** argument is a long integer.
pichgoal*N*	Desired height of the picture in twips. The ***N*** argument is a long integer.
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.
picprop	Indicates there are shape properties applied to an inline picture. This is a destination control word.
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.
blipupi*N*	***N*** represents units per inch on a picture (only certain image types need or output this)
blipuid XXXXX	Used as: `{\*\blipuid XXXXX}` where *XXXX* is a 16-byte identification number for the image.
bliptag*N*	A mostly unique identifier for a picture, where ***N*** is a long integer value.

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
02040000200010275023e000000000
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 | objhtml | objocx**

<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.
\objhtml	An object type of HTML control.
\objocx	An object type of OLE control.
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.

When Word is used as an editor for Mail, the following control word can be emitted. It is not seen in other situations.

Control Word	Meaning
\objattph	Object attachment placeholder. Used in the RTF stream when Word is started as a mail editor and the message contains attachments. The control word tells where in the text stream the attachment should be placed. It does not define the actual attachment.

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 41 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

Word 6.0/95 RTF

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**, whereas 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	Absolute distance-attached polyline.
\dpcodbottom	Bottom-attached polyline.
\dpcodcenter	Center-attached polyline.
\dpcodtop	Top-attached callout.
\dpcodescent*N*	The descent of the 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	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

Word 97 RTF for Drawing Objects (Shapes)

Basic Format

The basic format for drawing objects in RTF is as follows

```
{ \shp   ........  { \*\shpinst  { \spp  { \sn .......... }  { \sp .............. }  }  }
              { \shprslt   ............... }   }
```

The first destination (**\shp**) is always present. This control word groups everything related to a shape together. Following the destination change, comes basic information regarding the shape. The following keywords with values can appear in any order after the "{ \shp" control word.

Control word	Meaning
Shape keywords	
\shpleft*N*	The value ***N*** is a measurement in twips. Specifies position of shape from the left of the anchor.
\shptop*N*	The value ***N*** is a measurement in twips. Specifies position of shape from top of the anchor.
\shpbottom*N*	The value ***N*** is a measurement in twips. Specifies position of shape from bottom of the anchor.
\shpright*N*	The value ***N*** is a measurement in twips. Specifies position of shape from right of the anchor.
\shplid*N*	A number that is unique to each shape. This keyword is primarily used for linked text boxes. The value ***N*** is a long integer.
\shpz*N*	Describes z-order of shape. It starts at 0 for the back most shape and proceed to ***N*** for the top most shape. The shapes that appear inside of the header document will have a separate z-order as compared to the z-order of the shapes in the main document. For instance the back-most shape in the header will have z-order number 0, and the back-most main-document shape will also have z-order number 0.
\shpfhdr*N*	0 if the shape is in the main document. 1 if the shape is in the header document.
\shpbxpage	The shape is positioned relative to the page in the x (horizontal) direction.
\shpbxmargin	The shape is positioned relative to the margin in the x (horizontal) direction.
\shpbxcolumn	The shape is positioned relative to the column in the x (horizontal) direction.
\shpbypage	The shape is positioned relative to the page in the y (vertical) direction.
\shpbymargin	The shape is positioned relative to the margin in the y (vertical) direction.
\shpbypara	The shape is positioned relative to the paragraph in the y (vertical) direction.

\shpwr*N*	Describes the type of wrap for the shape. 1 Wrap around top and bottom of shape (no text allowed beside shape) 2 Wrap around shape 3 None (wrap as if shape isn't present) 4 Wrap tightly around shape 5 Wrap text through shape
\shpwrk*N*	Wrap on side (for types 2 and 4 for **\shpwr*N***). 0 Wrap both sides of shape 1 Wrap left side only 2 Wrap right side only 3 Wrap only on largest side
\shpfblwtxt*N*	Describes relative z-ordering. 0 Text is below shape 1 Shape is below text
\shplockanchor	Lock anchor for shape.
\shptxt	Text for a shape. The text must come after all the other properties for the shape (inside the \shpinst destination) in the following format: `{ \shptxt  Any Valid RTF for the current textbox }` **Note** For linked text boxes, the first text box of the linked set has the entire story, so all following text boxes will not have a \shptxt field.
\shprslt	This is where the Word 6.0/95 drawn object RTF can be placed.
\shpgrp	Specifies a group shape. The parameters following this keyword are the same as those following **\shp**. The order of the shapes inside a group is from bottom to top in z-order. Inside of a \shpgrp, no `{ \shprslt .... }` fields would be generated (that is, only the root-level shape can have a **\shprslt** field (this field describes the entire group). For example: `{ \shpgrp  ....... { \shp ..... (and all sub-items as usual) }` `{ \shp ......(and all sub-items as usual) }` **Note** A `{ \shpgrp ...... }` can be substituted for a `{ \shp ..... }` at any place (to accomplish groups inside of groups).

With the exception of **\shplid,** these do not apply for shapes that are within a group. For more information about groups, see the "Introduction" section of this Applicaiton Note.

Control word	**Meaning**
\background	Specifies the document background. This is a destination keyword. It contains the `{ \shp` keyword and all the shape properties.

Drawing Object Properties

The `{ \shp ...........` control word is followed by `{ \*\shpinst`

The bulk of a shape is defined as a series of properties. Following the `{ \*\shpinst` is a list of all the properties of a shape each in the following format:

```
{ \sp  { \sn PropertyName } { \sv PropertyValueInformation } }
```

The control word for the drawing object property is **\sp.** Each property has a pair of name (**\sn**) and value (**\sv**) control words placed in the shape property group. For example, the vertical flip property is represented as:

```
{\sp{\sn fFlipV}{\sv 1}}
```

Here, the name of the property is **fFlipV** and the value is 1, which indicates **True**. All shape properties follow this basic format. Only properties that have been explicitly set for a shape are written out in RTF format. Other properties assume the default values (a property may be set to the default value explicitly).

The following table describes all the names of properties for drawing objects along with the type of their corresponding value.

Property	Type of Value	Meaning
Object Type		
Rotation	Angle	Rotation of the shape.
FFlipV	Boolean	Vertical flip, applied after the rotation.
FFlipH	Boolean	Horizontal flip, applied after the rotation.
ShapeType		See below for values. 0 indicates user-drawn freeforms and polygons.
WzName	String	Shape name (only set through Visual Basic® for Applications).
pWrapPolygonVertices	Array	Points of the text wrap polygon.
dxWrapDistLeft	EMU	Left wrapping distance from text.
dyWrapDistTop	EMU	Top wrapping distance from text.
dxWrapDistRight	EMU	Right wrapping distance from text.
dyWrapDistBottom	EMU	Bottom wrapping distance from text.
fBehindDocument	Boolean	Place the shape behind text.
fIsButton	Boolean	Specified whether the shape is a button.
fHidden	Boolean	Do not display or print (only set through Visual Basic for Applications).
Lock		
fLockRotation	Boolean	Lock rotation.
fLockAspectRatio	Boolean	Lock aspect ratio.
fLockAgainstSelect	Boolean	No selecting this shape.
fLockCropping	Boolean	No cropping this shape.
fLockVerticies	Boolean	No points edit mode.

fLockText	Boolean	Do not edit text.
fLockAdjustHandles	Boolean	Do not adjust.
fLockAgainstGrouping	Boolean	Do not group this shape.
Text Box		
dxTextLeft	EMU	Left internal margin of the text box.
dyTextTop	EMU	Top internal margin of the text box.
dxTextRight	EMU	Right internal margin of the text box.
dyTextBottom	EMU	Bottom internal margin of the text box.
WrapText		Wrap text at shape margins: 0 Square 1 Tight 2 None 3 Top Bottom 4 Through
anchorText		Text anchor point: 0 Top 1 Middle 2 Bottom 3 Top Centered 4 Middle Centered 5 Bottom Centered 6 Bottom Centered Baseline
txflTextFlow		Text flow: 0 Horizontal non-ASCII font 1 Top to bottom ASCII font 2 Bottom to top non-ASCII font 3 Top to bottom non-ASCII font 4 Horizontal ASCII font
WordArt Effect		
gtextUNICODE	String	Unicode text string.
gtextAlign		Alignment on curve: 0 Stretch each line of text to fit width 1 Center text on width 2 Left justify 3 Right justify 4 Spread letters out to fit width 5 Spread words out to fit width
gtextSize	Fixed	Default point size.

gtextSpacing	Fixed	Adjust the spacing between characters (1.0 is normal).
gtextFont	String	Font name.
fGtext	Boolean	**True** if the text effect properties (**gtext***) are used.
		False if these properties are ignored.
gtextFVertical	Boolean	If an @ font is available use it; otherwise, rotate individual characters 90 degrees counter-clockwise.
gtextFKern	Boolean	If the font supports character pair kerning, use it.
gtextFTight	Boolean	Adjust the spacing between characters rather than the character advance by the **gtextSpacingratio.**
gtextFStretch	Boolean	Stretch the text to fit shape.
gtextFShrinkFit	Boolean	When laying out the characters, consider the glyph bounding box rather than the nominal font character bounds.
gtextFBestFit	Boolean	Scale text laid out on a path to fit the path.
gtextFNormalize	Boolean	Stretch individual character heights independently to fit.
gtextFDxMeasure	Boolean	When laying out characters, measure distances along the x-axis rather than along the path.
gtextFBold	Boolean	Bold font (if available).
gtextFItalic	Boolean	Italic font (if available).
gtextFUnderline	Boolean	Underline font (if available).
gtextFShadow	Boolean	Shadow font (if available).
gtextFSmallcaps	Boolean	Small caps font (if available).
gtextFStrikethrough	Boolean	Strikethrough font (if available).
Picture		
cropFromTop	Fixed	Top cropping percentage.
cropFromBottom	Fixed	Bottom cropping percentage.
cropFromLeft	Fixed	Left cropping percentage.
cropFromRight	Fixed	Right cropping percentage.
pib	Picture	Binary picture data.
pibName	String	Picture file name for link to file pictures.
pibFlags		Flags for linked to file pictures:
		0 No links (default)
		10 Link to file; save with document
		14 Link to file; do not save picture with document
pictureTransparent	Color	Transparent color.
pictureContrast	Fixed	Contrast setting.
PictureBrightness	Fixed	Brightness setting.
pictureGamma	Fixed	Gamma correction setting.
pictureGray	Boolean	Display grayscale.
pictureBiLevel	Boolean	Display bi-level.

Geometry		
geoLeft	Long integer	Left edge of the bounds of a user-drawn shape.
geoTop	Long integer	Top edge of the bounds of a user-drawn shape.
geoRight	Long integer	Right edge of the bounds of a user-drawn shape.
geoBottom	Long integer	Bottom edge of the bounds of a user-drawn shape.
pVerticies	Array	The points of the shape.
pSegmentInfo	Array	The segment information.
adjustValue	Integer	First adjust value from an adjust handle. The interpretation varies with the shape type. Adjust values alter the geometry of the shape in smart ways.
adjust2Value	Integer	Second adjust value.
adjust3Value	Integer	Third adjust value.
adjust4Value	Integer	Fourth adjust value.
adjust5Value	Integer	Fifth adjust value.
adjust6Value	Integer	Sixth adjust value.
adjust7Value	Integer	Seventh adjust value.
adjust8Value	Integer	Eighth adjust value.
adjust9Value	Integer	Ninth adjust value.
adjust10Value	Integer	Tenth adjust value.
Fill		
fillType	Fill type	Type of fill: 0 A solid color 1 A pattern (bitmap) 2 A texture (pattern with its own color map) 3 A picture centered in the shape 4 Shade from start to end points 5 Shade from bounding rectangle to end point 6 Shade from shape outline to end point 7 Shade using the fillAngle
fillColor	Color	Foreground color.
fillOpacity	Fixed	Opacity. Normal is 1.0.
fillBackColor	Color	Background color.
fillBackOpacity	Fixed	Opacity for shades only. Normal is 1.0.
fillBlip	Picture	Pattern/texture picture for the fill.
fillBlipName	String	Picture file name for custom fills.
fillblipflags		Flags for fills: 0 No links (default) 10 Link to file; save with document 14 Link to file; do not save picture with document

fillWidth	EMU	The pattern or tile will be expanded to approximately this size.
fillHeight	EMU	The pattern or tile will be expanded to approximately this size.
fillAngle	Fixed	Fade angle number of degrees.
fillFocus		Linear shaded fill focus percent.
fillToLeft	Fixed	The **fillToLeft**, **fillToTop**, **fillToRight**, and **fillToBottom** values define the "focus" rectangle for concentric shapes; they are specified as a fraction of the outer rectangle of the shade.
fillToTop	Fixed	See **fillToLeft** definition.
fillToRight	Fixed	See **fillToLeft** definition.
fillToBottom	Fixed	See **fillToLeft** definition.
fillShadeColors	Array	Custom or preset color ramps for graduated fills on shapes.
fillOriginX	Fixed	When a textured fill is used, the texture may be aligned to with shape (**fFillShape**)—if this is done, the default alignment is to the top left. The values **FillOriginY** **FillShapeOriginX** **fillShapeOriginY** allow an arbitrary position in the texture (relative to the top-left proportion of the texture's height and width) to be aligned on an arbitrary position on the shape (relative to the top-left proportion of the width and height of the bounding box). Note that all these values are fixed point fractions of the relevant width or height.
fillOriginY	Fixed	See **fillOriginX** definition.
fillShapeOriginX	Fixed	See **fillOriginX** definition.
fillShapeOriginY	Fixed	See **fillOriginX** definition.
fFilled	Boolean	The shape is filled.
Line		
lineColor	Color	Color of the line.
lineBackColor	Color	Background color of the pattern.
lineType	Line type	Type of line: 0 Solid fill with the line color 1 Patterned fill with the **lineFillBlip** 2 Textured fill with the **lineFillBlip** 3 Picture fill with the **lineFillBlip**
lineFillBlip	Picture	Pattern for the line.

lineblipflags		Flags for patterned lines: 0 No links (default) 10 Link to file; save with document 14 Link to file; do not save picture with document
lineFillWidth	EMU	Width of the pattern
lineFillHeight	EMU	Height of the pattern
lineWidth	EMU	Line width
lineStyle		Line style: 0 Single line (of width **lineWidth**) 1 Double lines of equal width 2 Double lines, one thick, one thin 3 Double lines, reverse order 4 Three lines, thin, thick, thin
lineDashing		Dashing: 0 Solid 1 Dash (Windows) 2 Dot (Windows) 3 Dash dot (Windows) 4 Dash dot dot (Windows) 6 Dot 7 Dash 8 Long dash 9 Dash dot 10 Long dash dot 11 Long dash dot dot
lineStartArrowhead		Start arrow type: 0 Nothing 1 Arrow 2 Stealth arrow 3 Diamond 4 Oval 6 Open arrow 7 Chevron arrow 8 Double chevron arrow
lineEndArrowhead		End arrow type (same values as for **lineStartArrowhead**).
lineStartArrowWidth		Start arrow width: 0 Narrow 1 Medium 2 Wide

lineStartArrowLength		Start arrow length: 0 Short 1 Medium 2 Long
lineEndArrowWidth		End arrow width (same values as for **lineStartArrowWidth**).
lineEndArrowLength		End arrow length (same values as for **lineStartArrowLength**).
fLine	Boolean	Has a line.
Shadow		
shadowType		Type of shadow: 0 Offset shadow 1 Double offset shadow 2 Rich perspective shadow (cast relative to shape) 3 Rich perspective shadow (cast in shape space) 4 Perspective shadow cast in drawing space 6 Emboss or engrave
shadowColor	Color	Foreground color.
shadowHighlight	Color	Embossed color.
shadowOpacity	Fixed	Opacity of the shadow. Normal is 1.0.
shadowOffsetX	EMU	Shadow offset toward the right.
shadowOffsetY	EMU	Shadow offset toward the bottom.
shadowSecondOffsetX	EMU	Double shadow offset toward the right.
shadowSecondOffsetY	EMU	Double shadow offset toward the bottom.
shadowScaleXToX	Fixed	The **shadowScaleXToX** to **shadowWeight** define a 3x2 transform matrix that is applied to the shape to generate the shadow.
shadowScaleYToX	Fixed	See definition for **shadowScaleXToX**.
shadowScaleXToY	Fixed	See definition for **shadowScaleXToX**.
shadowScaleYToY	Fixed	See definition for **shadowScaleXToX**.
shadowPerspectiveX	Fixed	See definition for **shadowScaleXToX**.
shadowPerspectiveY	Fixed	See definition for **shadowScaleXToX**.
shadowWeight	Fixed	See definition for **shadowScaleXToX**.
shadowOriginX	Fixed	Define the position of the origin relative to the center of the shape— this position is determined based on a proportion of the *rotated* shape width and height. The shape will be rotated and then positioned such that the point is at (0,0) before the transformation is applied.
ShadowOriginY	Fixed	See the definition for **shadowOriginX.**
fShadow	Boolean	Switches the shadow on or off.
3-D Effects		
c3DSpecularAmt	Fixed	Specular amount for the material.

c3DDiffuseAmt	Fixed	Diffusion amount for the material.
c3DShininess	Fixed	Shininess of the material.
c3DEdgeThickness	EMU	Specular edge thickness.
c3DExtrudeForward	EMU	Extrusion amount forward.
c3DExtrudeBackward	EMU	Extrusion amount backward.
c3DExtrusionColor	Color	Color of the extrusion.
f3D	Boolean	**True** if shape has a three-dimensional (3D) effect, **False** if it does not.
fc3DMetallic	Boolean	**True** if shape uses metallic specularity, **False** if it does not.
fc3DUseExtrusionColor	Boolean	Extrusion color is set explicitly.
fc3DLightFace	Boolean	Light the face of the shape.
c3DYRotationAngle	Angle	Degrees about y-axis. If **fc3DconstrainRotation** (a Boolean property which defaults to **True**) is **True** the rotation is restricted to x-y rotation and the final rotation results from first rotating by **c3DYRotationAngle** degrees about the y-axis and then by **c3DXRotationAngle** degrees about the z-axis. If **fc3DconstrainRotation** is **False**, the final rotation results from a single rotation of **c3DrotationAngle** about the axis specified by **c3DrotationAxisX**, **c3DrotationAxisY**, and **c3DrotationAxisZ**.
c3DXRotationAngle	Angle	Degrees about x-axis.
c3DRotationAxisX	Long integer	These specify the rotation axis. Only their relative magnitudes matter.
c3DRotationAxisY	Long integer	See the **c3DYRotationAxisX** definition.
c3DRotationAxisZ	Long integer	See the **c3DYRotationAxisX** definition.
c3DRotationAngle	Angle	The rotation about the axis (defined above in the **c3DRotationAxisX**, **Y,** and **Z** parameter sections)
fC3DRotationCenterAut	Boolean	If **fC3DRotationCenterAuto** is **True** the rotation will be about the center of the 3-D bounding cube of the 3-D group; otherwise, the rotation center will be about **c3DRotationCenterX**, **c3DRotationCenterY**, and **c3DRotationCenterZ**.
c3DRotationCenterX	Fixed	Rotation center (X). The **X** and **Y** values are a 16.16 fraction of the geometry width and height, with (0,0) being at the center of the geometry. The **Z** value must be in absolute units (EMUs).
c3DRotationCenterY	Fixed	Rotation center (Y). If **fC3DRotationCenterAuto** is **True** the rotation will be about the center of the 3-D bounding cube of the 3-D group; otherwise, the rotation center will be about **c3DRotationCenterX**, **c3DRotationCenterY**, and **c3DRotationCenterZ**. The **X** values and **Y** values are a fraction of the geometry width and height, with (0,0) being at the center of the geometry. The **Z** value is in absolute units.

c3DRotationCenterZ	EMU	See **c3DRotationCenterY** above.
c3DRenderMode	Long Integer	0 Render with full detail 1 Render as a wire frame 2 Render a bounding cube
c3DXViewpoint	EMU	X view point.
c3DYViewpoint	EMU	Y view point.
c3DZViewpoint	EMU	Z view distance.
c3DOriginX	Fixed	The following **c3DOriginY** and **c3DSkewAngle** values define the origin relative to which the viewpoint origin is measured. These values are 16.16 numbers that specify the position of the origin within the shape bounding box as multiples of the width and height of that bounding box and relative to the center (that is, they are displaced from the center). When these values are applied, the actual transformed shape path is used rather than the shape geometry (compare with the shadow and perspective values which necessarily work on the geometry bounding box not the actual points). This means that a shape that extends outside the geometry bounding box (such as a text effect) is handled "correctly" for the calculation of the 3-D origin.
c3DOriginY	Fixed	See the definition for **c3DOriginX.**
c3DSkewAngle	Fixed	Skew angle.
c3DSkewAmount	Fixed	Percentage skew amount.
c3DAmbientIntensity	Fixed	Ambient intensity should be low (0 to .1) to avoid washed out appearance.
c3DKeyX	Long integer	Key light source direction. Values may be any number; only their relative magnitudes matter.
c3DKeyY	Long integer	See **c3DKeyX** definition above.
c3DKeyZ	Long integer	See **c3DKeyX** definition above.
c3DKeyIntensity	Fixed	Fixed point intensity. Theoretical maximum is 1, but can be higher.
c3DFillX	Long integer	Fill light source direction; only their relative magnitudes matter. This direction defines a second light source arbitrarily called the "fill light." Generally this will be positioned 90-180 degrees away from the key light and very roughly in front of the scene to fill in any harsh shadows. This fill will be dim compared to the first light source. Theoretically it should be non-harsh, but harsh fill lighting looks better sometimes.
c3DFillY	Long integer	See **c3DfillX** definition.
c3DFillZ	Long integer	See **c3DfillX** definition.
c3DFillIntensity	Fixed	Theoretical maximum is 1, but can be higher.
fc3DParallel	Boolean	**True** if the fill has parallel projection, **False** if it does not. If fc3DParallel is **True**, the **fc3DKeyHarsh** and **fc3DFillHarsh** properties determine the parallel projection used. A skew amount of 0 means the projection is orthographic.

fc3DKeyHarsh	Boolean	**True** if key lighting is harsh, **False** if it is not.
fc3DFillHarsh	Boolean	**True** if fill lighting harsh, **False** if it is not.
Callout		
spcot		Callout type: 1 Right angle 2 One segment 3 Two segments 4 Three segments
dxyCalloutGap	EMU	Distance from box to first point.
spcoa		Callout angle: 1 Any angle 2 30 degrees 3 43 degrees 4 60 degrees 5 90 degrees
spcod		Callout drop type: 0 Top 1 Center 2 Bottom 3 Specified by dxyCalloutDropSpecified
dxyCalloutDropSpecified	EMU	If spcod is 3, then this holds the actual drop distance.
dxyCalloutLengthSpecified	EMU	In the case where fCalloutLengthSpecified is **True**, this holds the actual distance.
fCallout	Boolean	This is a callout.
fCalloutAccentBar	Boolean	Callout has an accent bar.
fCalloutTextBorder	Boolean	Callout has a text border.
fCalloutDropAuto	Boolean	**True** if Auto attach is on. **False** if it is off. If this is **True**, then the converter should occasionally invert the drop distance.
fCalloutLengthSpecified	Boolean	**True** if the callout length is specified; **False** if it is not. If **True**, use **dxyCalloutLengthSpecified.** If **False**, the **Best Fit** option is on.

The format of the value depends on the property name it is paired with. Many values are simple single numbers. Distances are expressed in EMU units. There are 12700 EMU units in a point hence 914400 in an inch and 360000cm^{-1}. Fractional or fixed values are expressed using units that are 1/65536th of a whole. Angles are expressed as fractions of a degree. Colors are 24 bit color values. Booleans have two possible values: 1 for **True** and 0 for **False**.

Arrays are formatted as a sequence of number separated by semicolons. The first number tells the size of each element in the array in bytes. The number of bytes per element may be 2, 4, or 8. When the size of the element is 8, each element is represented as a group of two numbers. The second number tells the number of elements in the array. For example, the points of a square polygon are written as:

```
{\sv 8;4;{0,0};{100,0};{100,100};{0,100}}
```

The **ShapeType** property can have the following possible values.

Value	Description
0	Freeform or non-autoshape
1	Rectangle
2	Round rectangle
3	Ellipse
4	Diamond
5	Isosceles triangle
6	Right triangle
7	Parallelogram
8	Trapezoid
9	Hexagon
10	Octagon
11	Plus Sign
12	Star
13	Arrow
14	Thick arrow
15	Home plate
16	Cube
17	Balloon
18	Seal
19	Arc
20	Line
21	Plaque
22	Can
23	Donut
24	Text simple
25	Text octagon
26	Text hexagon
27	Text curve
28	Text wave
29	Text ring
30	Text on curve
31	Text on ring
41	Callout 1
42	Callout 2
43	Callout 3
44	Accent Callout 1

45	Accent Callout 2
46	Accent Callout 3
47	Border Callout 1
48	Border Callout 2
49	Border Callout 3
50	Accent Border Callout 1
51	Accent Border Callout 2
52	Accent Border Callout 3
53	Ribbon
54	Ribbon2
55	Chevron
56	Pentagon
57	No Smoking
58	Seal8
59	Seal16
60	Seal32
61	Wedge Rect Callout
62	Wedge RRect Callout
63	Wedge Ellipse Callout
64	Wave
65	Folded Corner
66	Left Arrow
67	Down Arrow
68	Up Arrow
69	Left Right Arrow
70	Up Down Arrow
71	IrregularSeal1
72	IrregularSeal2
73	Lightning Bolt
74	Heart
75	Picture Frame
76	Quad Arrow
77	Left Arrow Callout
78	Right Arrow Callout
79	Up Arrow Callout
80	Down Arrow Callout
81	Left Right Arrow Callout
82	Up Down Arrow Callout
83	Quad Arrow Callout

84	Bevel
85	Left Bracket
86	Right Bracket
87	Left Brace
88	Right Brace
89	Left Up Arrow
90	Bent Up Arrow
91	Bent Arrow
92	Seal24
93	Striped Right Arrow
94	Notched Right Arrow
95	Block Arc
96	Smiley Face
97	Vertical Scroll
98	Horizontal Scroll
99	Circular Arrow
100	Notched Circular Arrow
101	Uturn Arrow
102	Curved Right Arrow
103	Curved Left Arrow
104	Curved Up Arrow
105	Curved Down Arrow
106	Cloud Callout
107	Ellipse Ribbon
108	Ellipse Ribbon 2
109	Flow Chart Process
110	Flow Chart Decision
111	Flow Chart Input Output
112	Flow Chart Predefined Process
113	Flow Chart Internal Storage
114	Flow Chart Document
115	Flow Chart Multidocument
116	Flow Chart Terminator
117	Flow Chart Preparation
118	Flow Chart Manual Input
119	Flow Chart Manual Operation
120	Flow Chart Connector
121	Flow Chart Punched Card
122	Flow Chart Punched Tape

123	Flow Chart Summing Junction
124	Flow Chart Or
125	Flow Chart Collate
126	Flow Chart Sort
127	Flow Chart extract
128	Flow Chart Merge
129	Flow Chart Offline Storage
130	Flow Chart Online Storage
131	Flow Chart Magnetic Tape
132	Flow Chart Magnetic Disk
133	Flow Chart Magnetic Drum
134	Flow Chart Display
135	Flow Chart Delay
136	Text Plain Text
137	Text Stop
138	Text Triangle
139	Text Triangle Inverted
140	Text Chevron
141	Text Chevron Inverted
142	Text Ring Inside
143	Text Ring Outside
144	Text Arch Up Curve
145	Text Arch Down Curve
146	Text Circle Curve
147	Text Button Curve
148	Text Arch Up Pour
149	Text Arch Down Pour
150	Text Circle Pour
151	Text Button Pour
152	Text Curve Up
153	Text Curve Down
154	Text Cascade Up
155	Text Cascade Down
156	Text Wave1
157	Text Wave2
158	Text Wave3
159	Text Wave4
160	Text Inflate
161	Text Deflate

162	Text Inflate Bottom
163	Text Deflate Bottom
164	Text Inflate Top
165	Text Deflate Top
166	Text Deflate Inflate
167	Text Deflate Inflate Deflate
168	Text Fade Right
169	Text Fade Left
170	Text Fade Up
171	Text Fade Down
172	Text Slant Up
173	Text Slant Down
174	Text Can Up
175	Text Can Down
176	Flow Chart Alternate Process
177	Flow Chart Off-Page Connector
178	Callout 90
179	Accent Callout 90
180	Border Callout 90
181	Accent Border Callout 90
182	Left Right Up Arrow
183	Sun
184	Moon
185	Bracket Pair
186	Brace Pair
187	Seal4
188	Double Wave
201	Host Control
202	Text Box

The following keywords are related to defining a hyperlink hanging off of a shape (that is, all of them are inside of a `{\sp {\sn … } {\sp …}}`). These specifically can occur in the \sp to define a property that is a hyperlink. They are used like this:

```
{ \hl  { \hlloc  RTF-string } { \hlsrc RTF-string } { \hlfr RTF-string } }
```

The three groups can be in any order. These provide the three strings needed to describe a hyperlink fully.

Control word	**Meaning**
Hyperlink property for shapes	
\hlloc	Location string for hyperlink.
\hlsrc	Source string for hyperlink.
\hlfr	Friendly name for hyperlink.

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 comments (annotations). Placing a footnote within headers, footers, or comments (annotations) will often result in a corrupted 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 16), "Section Formatting Properties" (page 20), and "Special Characters" (page 38) in this Application Note.

Comments (Annotations)

RTF comments (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 comments (annotations) within headers, footers, or footnotes. Placing an annotation within headers, footers, or footnotes will often result in a corrupted document. Each part of the annotation is an RTF destination. Comments (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*** '}'

Comments (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.}}
```

Comments (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 fields. Fields have the following syntax:

<field>	'{' **\field** <fieldmod>? <fieldinst> <fieldrslt> '}'
<fieldmod>	**\flddirty**? & **\fldedit**? & **\fldlock**? & **\fldpriv**?
<fieldinst>	'{*' **\fldinst** <para>+ <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 file name, then the **cpg** control can be used to define the character set of the file name. See “Code Page Support” on page 9 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. A field result should not start with a table, because this will break some RTF readers.

An example of some field text follows:

```
{\field {\*\fldinst AUTHOR \\*MERGEFORMAT     }{\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

000000000000000005546578743100084765652077768697a000000000000000000000}}}{\fldrslt Default
Result}}{\*\bkmkend Text1}
```

Note that the **datafield** destination requires the ***** prefix. The **fldtype**, **date**, **time,** and **wpeqn** field keywords should be ignored.

Form Fields

Control word	Meaning
\formfield	Group destination keyword indicating start of form field data.
\fftype*N*	Form field type:
	0 Text
	1 Check box
	2 List
\ffownhelp*N*	1 if there is associated Help text (defined under **ffhelptext**), 0 otherwise.
\ffownstat*N*	1 if there is associated status line text (defined under **ffstattext**), 0 otherwise.
\ffprot*N*	1 if this field is protected, 0 otherwise.
\ffsize*N*	Type of size selected for check box field:
	0 Auto
	1 Exact

\fftypetxt*N*	Type of text field: 0 Regular text 1 Number 2 Date 3 Current date 4 Current time 5 Calculation
\ffrecalc*N*	1 if the field should be calculated on exit, 0 otherwise.
\ffhaslistbox*N*	1 if this field has list box attached to it, 0 otherwise.
\ffmaxlen	Number of characters for text field.
\ffhps*N*	Check box size (half-point sizes).
\ffname	Form field name (string). This is a destination control word.
\ffdeftext	Default text for text field (string). This is a destination control word.
\ffdefres	Default entry for list field (for example 0 = first list item, 1 = second list item).
\ffformat	Format for text field (string). This is a destination control word.
\ffhelptext	Help text (string). This is a destination control word.
\ffstattext	Status line text (string). This is a destination control word.
\ffentrymcr	Macro to be executed upon entry into this form field (string). This is a destination control word.
\ffexitmcr	Macro to be executed upon exit from this form field (string). This is a destination control word.
\ffl	List of text for list field. This is a destination control word.
\ffres*N*	Result field for a form field. Values from 0 to ***N***-1, where ***N*** is the number of **ffl** entries.

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 34 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 "Associated Character Properties" on page 37 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 (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 (the default).
\rtlsect	This section will thread columns from right to left.
\ltrsect	This section will thread columns from left to right (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 (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.*

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, because 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 control words 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 braces that the reader has encountered in the RTF stream. When the reader sees an opening brace, it increments the count; when the reader sees a closing 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 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, comments (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 control words 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 control word. Once the reader isolates the RTF control word and its (possibly associated) value, the reader then searches an array of SYM structures to find the RTF control word. If the control word 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 control word 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.

Rtfactn.c

Rtfactn.c contains the tables describing the properties and control words, and the routines to evaluate properties (**ecApplyPropChange**) and to dispatch control words (**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 Control Word 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. Because **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 and later 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 are 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(b)) != 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

```
rtfreadr.exe: rtfactn.obj rtfreadr.obj
    link rtfreadr.obj rtfactn.obj <nul

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

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

APPENDIX B: WORD (ASIAN VERSIONS) TEXT FORMAT

This appendix contains the changes to the Rich Text Format (RTF) specification for the Japanese version of Word (all platforms). In this section, Word J refers to the Japanese version of Word 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 J.

This appendix is meant to be used in conjunction with the full RTF specification, assumes you have read the rest of this 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.

RTF-J

There is a Japanese local RTF specification, called RTF-J, that is somewhat different from the standard RTF specification. Although Word 7.0 J does not write RTF-J, it can read RTF-J files. It retains the text strings in the file and disregards unknown control words.

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 **HKEY_CURRENT_USERF\Software\Microsoft\Word\7.0FE**\ section of the registry database 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 J specifies the character set in the font table using **fcharset**. Word J interprets **cpg437** as **fcharset0** and **cpg932** as **fcharset128** if it encounters these control words 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 control words	Definition and the interpretation in Word
jis	RTF-J uses **jis** as a control word for character set. Word J 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 J interprets these as **fnil**, which is the default font family.

ShiftJIS Font Without \cpg or \fcharset

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

Composite Fonts (Associated Fonts for International Runs)

Word J 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 J.

<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 J 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 J will apply appropriate fonts to each character run in the style using the bulleted rules below.

Control word	Font that Word J 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 J 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

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

Control word	Description
Associated Character Properties	
\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.
\accnone	No accent characters (over dot / over comma).
\accdot	Over dot accent.
\acccomma	Over comma accent.
\charscalex	Character width scaling.
\striked	Double strikethrough.

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.
\dghspaceN	Grid horizontal spacing in twips (the default is 120).
\dgvspaceN	Grid vertical spacing in twips (the default is 120).
\dghoriginN	Grid horizontal origin in twips (the default is 1701).
\dgvoriginN	Grid vertical origin in twips (the default is 1984).
\dghshowN	Show ***N***th horizontal grid (the default is 3).
\dgvshowN	Show ***N***th vertical grid (the default is 0).
\twoonone	Print two logical pages on one physical page.

\lnongrid	Define line based on the grid.

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.

Frame Properties

\frmtxlrtb	Frame box flows from left to right and top to bottom (default).
\frmtxtbrl	Frame box flows right to left and top to bottom.
\frmtxbtlr	Frame box flows left to right and bottom to top.
\frmtxlrtbv	Frame box flows left to right and top to bottom, vertical.
\frmtxtbrlv	Frame box flows top to bottom and right to left, vertical.

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.
Table Formatting	
\clbrdrdtlcldglu	Diagonal line (top left to bottom right).
\clbrdrdtr\cldgll	Diagonal line (top right to bottom left).
\cltxlrtb	Text in a cell flows from left to right and top to bottom (default).
\cltxtbrl	Text in a cell flows right to left and top to bottom.
\cltxbtlr	Text in a cell flows left to right and bottom to top.
\cltxlrtbv	Text in a cell flows left to right and top to bottom, vertical.
\cltxtbrlv	Text in a cell flows top to bottom and right to left, vertical.
\clvmgf	The first cell in a range of table cells to be vertically merged.
\clvmrg	Contents of the table cell are vertically merged with those of the preceding cell.
\clvertalt	Cell top align.
\clvertalc	Cell vertically center align.
\clvertalb	Cell bottom align.
Tabs	
\tlmdot	Leader middle dots.

New Control Words Created by Asian Versions of Word 97

Control word	**Meaning**
Character Formatting Properties	
\cgrid*N*	Character grid.

\g	Destination related to character grids.
\gcw	Grid column width.
\gridtbl	Destination keyword related to character grids.
\nosectexpand	Disable character space basement.
Paragraph Formatting Properties	
\nosnaplinegrid	Disable snap line to grid.
\faauto	Font alignment the default setting for this is "Auto."
Borders	
\brdrframe	Border resembles a “Frame.”
Bullets and Numbers	
\pnaiueo	46 phonetic Katakana characters in "aiueo" order (*aiueo).
\pnaiueod	46 phonetic double-byte Katakana characters (*aiueo*dbchar).
\pndbnumd	Kanji numbering with the digit character (*dbnum2).
\pndbnumt	Kanji numbering 3 (*dbnum3).
\pndbnuml	Kanji numbering 3 (*dbnum3).
\pndbnumk	Kanji numbering 4 (*dbnum4).
\pnganada	Korean numbering 2 (*ganada).
\pngbnum	Chinese numbering 1 (*gb1).
\pngbnumd	Chinese numbering 2 (*gb2).
\pngbnuml	Chinese numbering 3 (*gb3).
\pngbnumk	Chinese numbering 4 (*gb4).
\pnzodiac	Chinese Zodiac numbering 1 (*zodiac1).
\pnzodiacd	Chinese Zodiac numbering 2 (*zodiac2).
\pnzodiacl	Chinese Zodiac numbering 3 (*zodiac3).
\pnganada	Korean numbering 1 (*ganada).
\pnchosung	Korean numbering 2 (*chosung).
Endnotes and Footnotes	
\ftnnchosung	Footnote Korean numbering 1 (*chosung).
\ftnncnum	Footnote Circle numbering (*circlenum).
\ftnndbnum	Foonote Kanji numbering without the digit character (*dbnum1).
\ftnndbnumd	Footnote Kanji numbering with the digit character (*dbnum2).
\ftnndbnumt	Footnote Kanji numbering 3 (*dbnum3).
\ftnndbnumk	Footnote Kanji numbering 4 (*dbnum4).
\ftnndbar	Footnote double-byte numbering (*dbchar).
\ftnnganada	Footnote Korean numbering 2 (*ganada).
\ftnngbnum	Footnote Chinese numbering 1 (*gb1).
\ftnngbnumd	Footnote Chinese numbering 2 (*gb2).
\ftnngbnuml	Footnote Chinese numbering 3 (*gb3).

\ftnngbnumk	Footnote Chinese numbering 4 (*gb4).
\ftnnzodiac	Footnote numbering— Chinese Zodiac numbering 1 (* zodiac1) 甲、乙、丙… 甲、乙、丙… 甲、乙、丙…
\ftnnzodiacd	Footnote numbering— Chinese Zodiac numbering 2 (* zodiac2) 子、丑、寅…
\ftnnzodiacl	Footnote numbering— Chinese Zodiac numbering 3 (* zodiac3).
\aftnnchosung	Endnote Korean numbering 1 (*chosung).
\aftnncnum	Endnote Circle numbering (*circlenum).
\aftnndbnum	Endnote Kanji numbering without the digit character (*dbnum1).
\aftnndbnumd	Endnote Kanji numbering with the digit character (*dbnum2).
\aftnndbnumt	Endnote Kanji numbering 3 (*dbnum3).
\aftnndbnumk	Endnote Kanji numbering 4 (*dbnum4).
\aftnndbar	Endnote double-byte numbering (*dbchar).
\aftnnganada	Endnote Korean numbering 2 (*ganada).
\aftnngbnum	Endnote Chinese numbering 1 (*gb1).
\aftnngbnumd	Endnote Chinese numbering 2 (*gb2).
\aftnngbnuml	Endnote Chinese numbering 3 (*gb3).
\aftnngbnumk	Endnote Chinese numbering 4 (*gb4).
\aftnnzodiac	Endnote numbering— Chinese Zodiac numbering 1 (* zodiac1) 甲、乙、丙…
\aftnnzodiacd	Endnote numbering— Chinese Zodiac numbering 2 (* zodiac2) 子、丑、寅…
\aftnnzodiacl	Endnote numbering— Chinese Zodiac numbering 3 (* zodiac3).

Section Formatting Properties

\pgnchosung	Korean numbering 1 (* chosung).
\pgncnum	Circle numbering (*circlenum).
\pgndbnumt	Kanji numbering 3 (*dbnum3).
\pgndbnumk	Kanji numbering 4 (*dbnum4).
\pgnganada	Korean numbering 2 (*ganada)
\pgngbnum	Chinese numbering 1 (*gb1).
\pgngbnumd	Chinese numbering 2 (*gb2).
\pgngbnuml	Chinese numbering 3 (*gb3).
\pgngbnumk	Chinese numbering 4 (*gb4).
\pgnzodiac	Chinese Zodiac numbering 1 (*zodiac1).
\pgnzodiacd	Chinese Zodiac numbering 2 (*zodiac2).
\pgnzodiacl	Chinese Zodiac numbering 3 (*zodiac3).
\sectexpand*N*	Character space basement.
\sectlinegrid*N*	Line grid.
\sectdefaultcl*N*	Default state of section. Indicates **\sectspecifycl*N*** and **\sectspecifyl*N*** are not emitted.
\sectspecifycl*N*	Specify number of characters per line only.
\sectspecifyl*N*	Specify both number of characters per line and number of lines per page.

adjustright	Adjust right indent.
Document Formatting Properties	
dgmargin	Grid to follow margins.
Index Entries	
yxe	Pronunciation for index entry.

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	This 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 7.0 are followed by an asterisk (*) and the names of all control words that are new to Microsoft Word 97 are followed by two asterisks (**).*

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	Associated Character Properties	Toggle
\absh	Positioned Objects and Frames	Value
\abslock *	Positioned Objects and Frames	Flag
\absw	Positioned Objects and Frames	Value
\acaps	Associated Character Properties	Toggle
\acf	Associated Character Properties	Value

\additive	Style Sheet	Flag
\adjustright **	New Control Words Created by Asian Versions of Word 97	Flag
\adn	Associated Character Properties	Value
\aenddoc	Document Formatting Properties	Flag
\aendnotes	Document Formatting Properties	Flag
\aexpnd	Associated Character Properties	Value
\af	Associated Character Properties	Value
\afs	Associated Character Properties	Value
\aftnbj	Document Formatting Properties	Flag
\aftncn	Document Formatting Properties	Destination
\aftnnalc	Document Formatting Properties	Flag
\aftnnar	Document Formatting Properties	Flag
\aftnnauc	Document Formatting Properties	Flag
\aftnnchi	Document Formatting Properties	Flag
\aftnnchosung **	New Control Words Created by Asian Versions of Word 97	Flag
\aftnncnum **	New Control Words Created by Asian Versions of Word 97	Flag
\aftnndbar **	New Control Words Created by Asian Versions of Word 97	Flag
\aftnndbnum **	New Control Words Created by Asian Versions of Word 97	Flag
\aftnndbnumd **	New Control Words Created by Asian Versions of Word 97	Flag
\aftnndbnumk **	New Control Words Created by Asian Versions of Word 97	Flag

\aftnndbnumt **	New Control Words Created by Asian Versions of Word 97	Flag
\aftnnganada **	New Control Words Created by Asian Versions of Word 97	Flag
\aftnngbnum **	New Control Words Created by Asian Versions of Word 97	Flag
\aftnngbnumd **	New Control Words Created by Asian Versions of Word 97	Flag
\aftnngbnumk **	New Control Words Created by Asian Versions of Word 97	Flag
\aftnngbnuml **	New Control Words Created by Asian Versions of Word 97	Flag
\aftnnrlc	Document Formatting Properties	Flag
\aftnnruc	Document Formatting Properties	Flag
\aftnnzodiac **	New Control Words Created by Asian Versions of Word 97	Flag
\aftnnzodiacd **	New Control Words Created by Asian Versions of Word 97	Flag
\aftnnzodiacl **	New Control Words Created by Asian Versions of Word 97	Flag
\aftnrestart	Document Formatting Properties	Flag
\aftnrstcont	Document Formatting Properties	Flag
\aftnsep	Document Formatting Properties	Destination
\aftnsepc	Document Formatting Properties	Destination
\aftnstart	Document Formatting Properties	Value
\aftntj	Document Formatting Properties	Flag
\ai	Associated Character Properties	Toggle
\alang	Associated Character Properties	Value

\allprot	Document Formatting Properties	Flag
\alt	Style Sheet	Flag
\animtextN **	New Control Words Created by Asian Versions of Word 97	Value
\annotation	Comments (annotations)	Destination
\annotprot	Document Formatting Properties	Flag
\ansi	Character Set	Flag
\ansicpgN **	Unicode RTF	Value
\aoutl	Associated Character Properties	Toggle
\ascaps	Associated Character Properties	Toggle
\ashad	Associated Character Properties	Toggle
\astrike	Associated Character Properties	Toggle
\atnauthor	Comments (annotations)	Destination
\atnicn	Comments (annotations)	Destination
\atnid	Comments (annotations)	Destination
\atnref	Comments (annotations)	Destination
\atntime	Comments (annotations)	Destination
\atrfend	Comments (annotations)	Destination
\atrfstart	Comments (annotations)	Destination
\aul	Associated Character Properties	Toggle
\auld	Associated Character Properties	Toggle
\auldb	Associated Character Properties	Toggle
\aulnone	Associated Character Properties	Toggle
\aulw	Associated Character Properties	Toggle
\aup	Associated Character Properties	Value
\author	Information Group	Destination
\b	Character Formatting Properties	Toggle
\background **	Word 97 RTF for Drawing Objects (Shapes)	Destination

\bdbfhdr **	Document Formatting Properties	Flag
\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
\bliptagN **	Pictures	Value
\blipuid **	Pictures	Value
\blipupiN **	Pictures	Value
\blue	Color Table	Value
\box	Paragraph Borders	Flag
\brdrartN **	Document Formatting Properties	Value
\brdrb	Paragraph Borders	Flag
\brdrbar	Paragraph Borders	Flag
\brdrbtw	Paragraph Borders	Flag
\brdrcf	Paragraph Borders	Value
\brdrdash	Paragraph Borders	Flag
\brdrdashd **	Paragraph Text	Flag

\brdrdashdd **	Paragraph Text	Flag
\brdrdashdotstr **	Paragraph Text	Flag
\brdrdashsm **	Paragraph Text	Flag
\brdrdb	Paragraph Borders	Flag
\brdrdot	Paragraph Borders	Flag
\brdremboss **	Paragraph Text	Flag
\brdrengrave **	Paragraph Text	Flag
\brdrframe **	New Control Words Created by Asian Versions of Word 97	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
\brdrthtnlg **	Paragraph Text	Flag
\brdrthtnmg **	Paragraph Text	Flag
\brdrthtnsg **	Paragraph Text	Flag
\brdrtnthlg **	Paragraph Text	Flag
\brdrtnthmg **	Paragraph Text	Flag
\brdrtnthsg **	Paragraph Text	Flag
\brdrtnthtnlg **	Paragraph Text	Flag
\brdrtnthtnmg **	Paragraph Text	Flag
\brdrtnthtnsg **	Paragraph Text	Flag
\brdrtriple **	Paragraph Text	Flag
\brdrw	Paragraph Borders	Value
\brdrwavy **	Paragraph Text	Flag
\brdrwavydb **	Paragraph Text	Flag
\brkfrm	Document Formatting Properties	Flag
\brsp	Paragraph Borders	Value
\bullet	Special Characters	Symbol
\buptim	Information Group	Destination
\bxe	Index Entries	Flag
\caps	Character Formatting Properties	Toggle

\category *	Information Group	Destination
\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
\cgridN **	New Control Words Created by Asian Versions of Word 97	Value
\charscalexN **	Character Text	Value
\chatn	Special Characters	Symbol
\chbgbdiag **	Character Text	Flag
\chbgcross **	Character Text	Flag
\chbgdcross **	Character Text	Flag
\chbgdkbdiag **	Character Text	Flag
\chbgdkcross **	Character Text	Flag
\chbgdkdcross **	Character Text	Flag
\chbgdkfdiag **	Character Text	Flag
\chbgdkhoriz **	Character Text	Flag
\chbgdkvert **	Character Text	Flag
\chbgfdiag **	Character Text	Flag
\chbghoriz **	Character Text	Flag
\chbgvert **	Character Text	Flag
\chbrdr **	Character Text	Flag
\chcbpatN **	Character Text	Value
\chcfpatN **	Character Text	Value
\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

\chshdngN **	Character Text	Value
\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
\cltxlrtb **	Paragraph Text	Flag
\cltxtbrl **	Paragraph Text	Flag
\clvertalb **	Paragraph Text	Flag
\clvertalc **	Paragraph Text	Flag
\clvertalt **	Paragraph Text	Flag
\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
\company *	Information Group	Destination
\cpg	Code Page Support	Value
\crauthN **	Character Text	Value
\crdateN **	Character Text	Value
\creatim	Information Group	Destination
\cs	Character Formatting Properties	Value
\ctrl	Style Sheet	Flag
\cvmme	Document Formatting Properties	Flag
\datafield	Fields	Destination
\date **	Fields	Flag
\deff	Font Table	Value
\defformat	Document Formatting Properties	Flag
\deflang	Document Formatting Properties	Value
\deflangfe **	Document Formatting Properties	Value
\deftab	Document Formatting Properties	Value
\deleted	Character Formatting Properties	Toggle
\dfrauthN **	Paragraph Text	Value
\dfrdateN **	Paragraph Text	Value
\dfrmtxtx	Positioned Objects and Frames	Value
\dfrmtxty	Positioned Objects and Frames	Value
\dfrstart **	Paragraph Text	Value
\dfrstop **	Paragraph Text	Value
\dfrxst **	Paragraph Text	Value
\dgmargin **	New Control Words Created by Asian Versions of Word 97	Flag
\dibitmap	Pictures	Value
\dn	Character Formatting Properties	Value

**\dntblnsbdb ** **	Document Formatting Properties	Flag
\do	Drawing Objects	Destination
\dobxcolumn	Drawing Objects	Flag
\dobxmargin	Drawing Objects	Flag
\dobxpage	Drawing Objects	Flag
\dobymargin	Drawing Objects	Flag
\dobypage	Drawing Objects	Flag
\dobypara	Drawing Objects	Flag
\doccomm	Information Group	Destination
\doctemp	Document Formatting Properties	Flag
**\doctypeN ** **	Document Formatting Properties	Value
\docvar *	Document Variables	Destination
\dodhgt	Drawing Objects	Value
\dolock	Drawing Objects	Flag
\dpaendhol	Drawing Objects	Flag
\dpaendl	Drawing Objects	Value
\dpaendsol	Drawing Objects	Flag
\dpaendw	Drawing Objects	Value
\dparc	Drawing Objects	Flag
\dparcflipx	Drawing Objects	Flag
\dparcflipy	Drawing Objects	Flag
\dpastarthol	Drawing Objects	Flag
\dpastartl	Drawing Objects	Value
\dpastartsol	Drawing Objects	Flag
\dpastartw	Drawing Objects	Value
\dpcallout	Drawing Objects	Flag
\dpcoa	Drawing Objects	Value
\dpcoaccent	Drawing Objects	Flag
\dpcobestfit	Drawing Objects	Flag
\dpcoborder	Drawing Objects	Flag
\dpcodabs	Drawing Objects	Value
\dpcodbottom	Drawing Objects	Flag
\dpcodcenter	Drawing Objects	Flag
\dpcodescent	Drawing Objects	Value
\dpcodtop	Drawing Objects	Flag

\dpcolength	Drawing Objects	Value
\dpcominusx	Drawing Objects	Flag
\dpcominusy	Drawing Objects	Flag
\dpcooffset	Drawing Objects	Value
\dpcosmarta	Drawing Objects	Flag
\dpcotdouble	Drawing Objects	Flag
\dpcotright	Drawing Objects	Flag
\dpcotsingle	Drawing Objects	Flag
\dpcottriple	Drawing Objects	Flag
\dpcount	Drawing Objects	Value
\dpellipse	Drawing Objects	Flag
\dpendgroup	Drawing Objects	Flag
\dpfillbgcb	Drawing Objects	Value
\dpfillbgcg	Drawing Objects	Value
\dpfillbgcr	Drawing Objects	Value
\dpfillbggray	Drawing Objects	Value
\dpfillbgpal	Drawing Objects	Flag
\dpfillfgcb	Drawing Objects	Value
\dpfillfgcg	Drawing Objects	Value
\dpfillfgcr	Drawing Objects	Value
\dpfillfggray	Drawing Objects	Value
\dpfillfgpal	Drawing Objects	Flag
\dpfillpat	Drawing Objects	Value
\dpgroup	Drawing Objects	Flag
\dpline	Drawing Objects	Flag
\dplinecob	Drawing Objects	Value
\dplinecog	Drawing Objects	Value
\dplinecor	Drawing Objects	Value
\dplinedado	Drawing Objects	Flag
\dplinedadodo	Drawing Objects	Flag
\dplinedash	Drawing Objects	Flag
\dplinedot	Drawing Objects	Flag
\dplinegray	Drawing Objects	Value
\dplinehollow	Drawing Objects	Flag
\dplinepal	Drawing Objects	Flag
\dplinesolid	Drawing Objects	Flag

\dplinew	Drawing Objects	Value
\dppolycount	Drawing Objects	Value
\dppolygon	Drawing Objects	Flag
\dppolyline	Drawing Objects	Flag
\dpptx	Drawing Objects	Value
\dppty	Drawing Objects	Value
\dprect	Drawing Objects	Flag
\dproundr	Drawing Objects	Flag
\dpshadow	Drawing Objects	Flag
\dpshadx	Drawing Objects	Value
\dpshady	Drawing Objects	Value
\dptxbx	Drawing Objects	Flag
\dptxbxmar	Drawing Objects	Value
\dptxbxtext	Drawing Objects	Destination
\dpx	Drawing Objects	Value
\dpxsize	Drawing Objects	Value
\dpy	Drawing Objects	Value
\dpysize	Drawing Objects	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
\embo **	Character Text	Toggle
\emdash	Special Characters	Symbol
\emfblip **	Pictures	Flag
\emspace	Special Characters	Symbol
\endash	Special Characters	Symbol
\enddoc	Document Formatting Properties	Flag
\endnhere	Section Formatting Properties	Flag
\endnotes	Document Formatting Properties	Flag

\enspace	Special Characters	Symbol
\expnd	Character Formatting Properties	Value
\expndtw	Character Formatting Properties	Value
\expshrtn **	Document Formatting Properties	Flag
\f	Character Formatting Properties	Value
\faauto **	New Control Words Created by Asian Versions of Word 97	Value
\facingp	Document Formatting Properties	Flag
\falt	Font Table	Destination
\fbiasN **	Font Table	Value
\fbidi	Font Table	Flag
\fcharset	Font Table	Value
\fdecor	Font Table	Flag
\fet	Document Formatting Properties	Value
\ffdefres **	Form Fields	Value
\ffdeftext **	Form Fields	Destination
\ffentrymcr **	Form Fields	Destination
\ffexitmcr **	Form Fields	Destination
\ffformat **	Form Fields	Destination
\ffhaslistboxN **	Form Fields	Value
\ffhelptext **	Form Fields	Destination
\ffhpsN **	Form Fields	Value
\ffl **	Form Fields	Destination
\ffmaxlen **	Form Fields	Value
\ffname **	Form Fields	Destination
\ffownhelpN **	Form Fields	Value
\ffownstatN **	Form Fields	Value
\ffprotN **	Form Fields	Value
\ffrecalcN **	Form Fields	Value
\ffresN **	Form Fields	Value
\ffsizeN **	Form Fields	Value
\ffstattext **	Form Fields	Destination
\fftypeN **	Form Fields	Value

\fftypetxtN **	Form Fields	Value
\fi	Paragraph Formatting Properties	Value
\fid	File Table	Value
\field	Fields	Destination
\file	File Table	Destination
\filetbl	File Table	Destination
\fldalt	Document Formatting Properties	Flag
\flddirty	Fields	Flag
\fldedit	Fields	Flag
\fldinst	Fields	Destination
\fldlock	Fields	Flag
\fldpriv	Fields	Flag
\fldrslt	Fields	Destination
\fldtype **	Fields	Destination
\fmodern	Font Table	Flag
\fn	Style Sheet	Value
\fname *	Font Table	Destination
\fnetwork	File Table	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	Document Formatting Properties	Flag
\formfield **	Form Fields	Destination
\formprot	Document Formatting Properties	Flag
\formshade	Document Formatting Properties	Flag
\fosnum	File Table	Value

\fprq	Font Table	Value
\fracwidth	Document Formatting Properties	Flag
\frelative	File Table	Value
\froman	Font Table	Flag
\fromtext **	Document Formatting Properties	Flag
\fs	Character Formatting Properties	Value
\fscript	Font Table	Flag
\fswiss	Font Table	Flag
\ftech	Font Table	Flag
\ftnalt	Document Formatting Properties	Flag
\ftnbj	Document Formatting Properties	Flag
\ftncn	Document Formatting Properties	Destination
\ftnil	Font Table	Flag
\ftnnalc	Document Formatting Properties	Flag
\ftnnar	Document Formatting Properties	Flag
\ftnnauc	Document Formatting Properties	Flag
\ftnnchi	Document Formatting Properties	Flag
\ftnnchosung **	New Control Words Created by Asian Versions of Word 97	Flag
\ftnncnum **	New Control Words Created by Asian Versions of Word 97	Flag
\ftnndbar **	New Control Words Created by Asian Versions of Word 97	Flag
\ftnndbnum **	New Control Words Created by Asian Versions of Word 97	Flag
\ftnndbnumd **	New Control Words Created by Asian Versions of Word 97	Flag
\ftnndbnumk **	New Control Words Created by Asian Versions of Word 97	Flag

\ftnndbnumt **	New Control Words Created by Asian Versions of Word 97	Flag
\ftnnganada **	New Control Words Created by Asian Versions of Word 97	Flag
\ftnngbnum **	New Control Words Created by Asian Versions of Word 97	Flag
\ftnngbnumd **	New Control Words Created by Asian Versions of Word 97	Flag
\ftnngbnumk **	New Control Words Created by Asian Versions of Word 97	Flag
\ftnngbnuml **	New Control Words Created by Asian Versions of Word 97	Flag
\ftnnrlc	Document Formatting Properties	Flag
\ftnnruc	Document Formatting Properties	Flag
\ftnnzodiac **	New Control Words Created by Asian Versions of Word 97	Flag
\ftnnzodiacd **	New Control Words Created by Asian Versions of Word 97	Flag
\ftnnzodiacl **	New Control Words Created by Asian Versions of Word 97	Flag
\ftnrestart	Document Formatting Properties	Flag
\ftnrstcont	Document Formatting Properties	Flag
\ftnrstpg	Document Formatting Properties	Flag
\ftnsep	Document Formatting Properties	Destination
\ftnsepc	Document Formatting Properties	Destination
\ftnstart	Document Formatting Properties	Value
\ftntj	Document Formatting Properties	Flag
\fttruetype	Font Table	Flag
\fvaliddos	File Table	Flag

\fvalidhpfs	File Table	Flag
\fvalidmac	File Table	Flag
\fvalidntfs	File Table	Flag
\g **	New Control Words Created by Asian Versions of Word 97	Destination
\gcw **	New Control Words Created by Asian Versions of Word 97	Value
\green	Color Table	Value
\gridtbl **	New Control Words Created by Asian Versions of Word 97	Destination
\gutter	Document Formatting Properties	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
\highlight *	Highlighting	Value
\hlfr **	Word 97 RTF for Drawing Objects (Shapes)	Value
\hlinkbase **	Information Group	Value
\hlloc **	Word 97 RTF for Drawing Objects (Shapes)	Value
\hlsrc **	Word 97 RTF for Drawing Objects (Shapes)	Value
\hr	Information Group	Value
\hyphauto	Document Formatting Properties	Toggle
\hyphcaps	Document Formatting Properties	Toggle
\hyphconsec	Document Formatting Properties	Value
\hyphhotz	Document Formatting Properties	Value
\hyphpar	Paragraph Formatting Properties	Toggle

\i	Character Formatting Properties	Toggle
\id	Information Group	Value
\ilvl **	Paragraph Text	Value
\impr **	Character Text	Toggle
\info	Information Group	Destination
\intbl	Paragraph Formatting Properties	Flag
\ixe	Index Entries	Flag
\jpegblip **	Pictures	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	Document Formatting Properties	Flag
\lang	Character Formatting Properties	Value
\ldblquote	Special Characters	Symbol
\level	Paragraph Formatting Properties	Value
\levelfollowN **	List Table	Value
\levelindentN **	List Table	Value
\leveljcN **	List Table	Value
\levellegalN **	List Table	Value
\levelnfcN **	List Table	Value
\levelnorestartN **	List Table	Value
\levelnumbers **	List Table	Destination
\leveloldN **	List Table	Value
\levelprevN **	List Table	Value
\levelprevspaceN **	List Table	Value
\levelspaceN **	List Table	Value
\levelstartatN **	List Table	Value
\leveltext **	List Table	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	Document Formatting Properties	Value
\linestarts	Section Formatting Properties	Value
\linex	Section Formatting Properties	Value
\linkself	Objects	Flag
\linkstyles	Document Formatting Properties	Flag
\linkval *	Information Group	Value
\listidN **	List Table	Value
\listname **	List Table	Destination
\listoverridecountN **	List Table	Value
\listoverrideformatN **	List Table	Value
\listoverridestartN **	List Table	Value
\listrestarthdnN **	List Table	Value
\listsimpleN **	List Table	Value
\listtemplateidN **	List Table	Value
\listtext **	Paragraph Text	Destination
\lndscpsxn	Section Formatting Properties	Flag
\lquote	Special Characters	Symbol
\ls **	List Table	Value
\ltrch	Character Formatting Properties	Flag
\ltrdoc	Document Formatting Properties	Flag
\ltrmark	Special Characters	Symbol
\ltrpar	Paragraph Formatting Properties	Flag
\ltrrow	Table Definitions	Flag

\ltrsect	Section Formatting Properties	Flag
\lytexcttp **	Document Formatting Properties	Flag
\lytprtmet **	Document Formatting Properties	Flag
\mac	Character Set	Flag
\macpict	Pictures	Flag
\makebackup	Document Formatting Properties	Flag
\manager *	Information Group	Destination
\margb	Document Formatting Properties	Value
\margbsxn	Section Formatting Properties	Value
\margl	Document Formatting Properties	Value
\marglsxn	Section Formatting Properties	Value
\margmirror	Document Formatting Properties	Flag
\margr	Document Formatting Properties	Value
\margrsxn	Section Formatting Properties	Value
\margt	Document Formatting Properties	Value
\margtsxn	Section Formatting Properties	Value
\min	Information Group	Value
\mo	Information Group	Value
\msmcap **	Document Formatting Properties	Flag
\nextfile	Document Formatting Properties	Destination
\nocolbal	Document Formatting Properties	Flag
\noextrasprl	Document Formatting Properties	Flag
\nofchars	Information Group	Value
\nofcharsws **	Information Group	Value
\nofpages	Information Group	Value
\nofwords	Information Group	Value

\nolead **	Document Formatting Properties	Flag
\noline	Paragraph Formatting Properties	Flag
\nonshppict **	Pictures	Flag
\nosectexpand **	New Control Words Created by Asian Versions of Word 97	Flag
\nosnaplinegrid **	New Control Words Created by Asian Versions of Word 97	Flag
\nospaceforul **	Document Formatting Properties	Flag
\nosupersub	Character Formatting Properties	Flag
\notabind	Document Formatting Properties	Flag
\noultrlspc **	Document Formatting Properties	Flag
\nowidctlpar	Paragraph Formatting Properties	Flag
\nowrap	Positioned Objects and Frames	Flag
\noxlattoyen **	Document Formatting Properties	Flag
\objalias	Objects	Destination
\objalign	Objects	Value
\objattph *	Objects	Flag
\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
\objhtml **	Objects	Flag
\objicemb	Objects	Flag
\objlink	Objects	Flag
\objlock	Objects	Flag

\objname	Objects	Destination
\objocx **	Objects	Flag
\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
\oldlinewrap **	Document Formatting Properties	Flag
\operator	Information Group	Destination
\otblrul	Document Formatting Properties	Flag
\outl	Character Formatting Properties	Toggle
\outlinelevelN **	Paragraph Text	Value
\overlay **	Paragraph Text	Flag
\page	Special Characters	Symbol
\pagebb	Paragraph Formatting Properties	Flag
\panose **	Font Table	Destination
\paperh	Document Formatting Properties	Value
\paperw	Document Formatting Properties	Value
\par	Special Characters	Symbol
\pard	Paragraph Formatting Properties	Flag
\pc	Character Set	Flag
\pca	Character Set	Flag
\pgbrdrb **	Document Formatting Properties	Flag
\pgbrdrfoot **	Document Formatting Properties	Flag
\pgbrdrhead **	Document Formatting Properties	Flag

\pgbrdrl **	Document Formatting Properties	Flag
\pgbrdroptN **	Document Formatting Properties	Value
\pgbrdrr **	Document Formatting Properties	Flag
\pgbrdrsnap **	Document Formatting Properties	Flag
\pgbrdrt **	Document Formatting Properties	Flag
\pghsxn	Section Formatting Properties	Value
\pgnchosung **	New Control Words Created by Asian Versions of Word 97	Flag
\pgncnum **	New Control Words Created by Asian Versions of Word 97	Flag
\pgncont	Section Formatting Properties	Flag
\pgndbnumk **	New Control Words Created by Asian Versions of Word 97	Flag
\pgndbnumt **	New Control Words Created by Asian Versions of Word 97	Flag
\pgndec	Section Formatting Properties	Flag
\pgnganada **	New Control Words Created by Asian Versions of Word 97	Flag
\pgngbnum **	New Control Words Created by Asian Versions of Word 97	Flag
\pgngbnumd **	New Control Words Created by Asian Versions of Word 97	Flag
\pgngbnumk **	New Control Words Created by Asian Versions of Word 97	Flag
\pgngbnuml **	New Control Words Created by Asian Versions of Word 97	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	Document Formatting Properties	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
\pgnzodiac **	New Control Words Created by Asian Versions of Word 97	Flag
\pgnzodiacd **	New Control Words Created by Asian Versions of Word 97	Flag
\pgnzodiacl **	New Control Words Created by Asian Versions of Word 97	Flag
\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
\picprop **	Pictures	Destination
\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	Bullets and Numbering	Destination
\pnacross	Bullets and Numbering	Flag
\pnaiueo **	New Control Words Created by Asian Versions of Word 97	Flag
\pnaiueod **	New Control Words Created by Asian Versions of Word 97	Flag
\pnb	Bullets and Numbering	Toggle
\pncaps	Bullets and Numbering	Toggle
\pncard	Bullets and Numbering	Flag
\pncf	Bullets and Numbering	Value
\pnchosung **	New Control Words Created by Asian Versions of Word 97	Flag
\pndbnumd **	New Control Words Created by Asian Versions of Word 97	Flag
\pndbnumk **	New Control Words Created by Asian Versions of Word 97	Flag
\pndbnuml **	New Control Words Created by Asian Versions of Word 97	Flag

\pndbnumt **	New Control Words Created by Asian Versions of Word 97	Flag
\pndec	Bullets and Numbering	Flag
\pnf	Bullets and Numbering	Value
\pnfs	Bullets and Numbering	Value
\pnganada **	New Control Words Created by Asian Versions of Word 97	Flag
\pnganada **	New Control Words Created by Asian Versions of Word 97	Flag
\pngblip **	Pictures	Flag
\pngbnum **	New Control Words Created by Asian Versions of Word 97	Flag
\pngbnumd **	New Control Words Created by Asian Versions of Word 97	Flag
\pngbnumk **	New Control Words Created by Asian Versions of Word 97	Flag
\pngbnuml **	New Control Words Created by Asian Versions of Word 97	Flag
\pnhang	Bullets and Numbering	Flag
\pni	Bullets and Numbering	Toggle
\pnindent	Bullets and Numbering	Value
\pnlcltr	Bullets and Numbering	Flag
\pnlcrm	Bullets and Numbering	Flag
\pnlvl	Bullets and Numbering	Value
\pnlvlblt	Bullets and Numbering	Flag
\pnlvlbody	Bullets and Numbering	Flag
\pnlvlcont	Bullets and Numbering	Flag
\pnnumonce	Bullets and Numbering	Flag
\pnord	Bullets and Numbering	Flag
\pnordt	Bullets and Numbering	Flag
\pnprev	Bullets and Numbering	Flag
\pnqc	Bullets and Numbering	Flag
\pnql	Bullets and Numbering	Flag
\pnqr	Bullets and Numbering	Flag
\pnrauthN **	Paragraph Text	Value

\pnrdateN **	Paragraph Text	Value
\pnrestart	Bullets and Numbering	Flag
\pnrnfcN **	Paragraph Text	Value
\pnrnot **	Paragraph Text	Flag
\pnrpnbrN **	Paragraph Text	Value
\pnrrgbN **	Paragraph Text	Value
\pnrstartN **	Paragraph Text	Value
\pnrstopN **	Paragraph Text	Value
\pnrxstN **	Paragraph Text	Value
\pnscaps	Bullets and Numbering	Toggle
\pnseclvl	Bullets and Numbering	Destination
\pnsp	Bullets and Numbering	Value
\pnstart	Bullets and Numbering	Value
\pnstrike	Bullets and Numbering	Toggle
\pntext	Bullets and Numbering	Destination
\pntxta	Bullets and Numbering	Destination
\pntxtb	Bullets and Numbering	Destination
\pnucltr	Bullets and Numbering	Flag
\pnucrm	Bullets and Numbering	Flag
\pnul	Bullets and Numbering	Toggle
\pnuld	Bullets and Numbering	Flag
\pnuldb	Bullets and Numbering	Flag
\pnulnone	Bullets and Numbering	Flag
\pnulw	Bullets and Numbering	Flag
\pnzodiac **	New Control Words Created by Asian Versions of Word 97	Flag
\pnzodiacd **	New Control Words Created by Asian Versions of Word 97	Flag
\pnzodiacl **	New Control Words Created by Asian Versions of Word 97	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
\posyin **	Paragraph Text	Flag
\posyout **	Paragraph Text	Flag
\posyt	Positioned Objects and Frames	Flag
\prcolbl	Document Formatting Properties	Flag
\printdata	Document Formatting Properties	Flag
\printim	Information Group	Destination
\private **	Document Formatting Properties	Destination
\propname *	Information Group	Value
\proptype *	Information Group	Value
\psover	Document Formatting Properties	Flag
\psz	Document Formatting Properties	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
\revauthdelN **	Character Text	Value
\revbar	Document Formatting Properties	Value
\revdttm	Character Formatting Properties	Value
\revdttmdelN **	Character Text	Value
\revised	Character Formatting Properties	Toggle
\revisions	Document Formatting Properties	Flag
\revprop	Document Formatting Properties	Value
\revprot	Document Formatting Properties	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	Document Formatting Properties	Flag
\rtlmark	Bidirectional Language Support and Special Characters	Symbol
\rtlpar	Paragraph Formatting Properties	Flag
\rtlrow	Table Definitions	Flag
\rtlsect	Section Formatting Properties	Flag
\rxe	Index Entries	Destination
\s	Paragraph Formatting Properties	Value
\sa	Paragraph Formatting Properties	Value
\sautoupd **	Style Sheet	Flag
\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
\sectdefaultclN **	New Control Words Created by Asian Versions of Word 97	Value
\sectexpandN **	New Control Words Created by Asian Versions of Word 97	Value

\sectlinegridN **	New Control Words Created by Asian Versions of Word 97	Value
\sectnum	Special Characters	Symbol
\sectspecifyclN **	New Control Words Created by Asian Versions of Word 97	Value
\sectspecifylN **	New Control Words Created by Asian Versions of Word 97	Value
\sectunlocked	Section Formatting Properties	Flag
\shad	Character Formatting Properties	Toggle
\shading	Paragraph Shading	Value
\shidden **	Style Sheet	Flag
\shift	Style Sheet	Flag
\shpbottomN **	Word 97 RTF for Drawing Objects (Shapes)	Value
\shpbxcolumn **	Word 97 RTF for Drawing Objects (Shapes)	Flag
\shpbxmargin **	Word 97 RTF for Drawing Objects (Shapes)	Flag
\shpbxpage **	Word 97 RTF for Drawing Objects (Shapes)	Flag
\shpbymargin **	Word 97 RTF for Drawing Objects (Shapes)	Flag
\shpbypage **	Word 97 RTF for Drawing Objects (Shapes)	Flag
\shpbypara **	Word 97 RTF for Drawing Objects (Shapes)	Flag
\shpfblwtxtN **	Word 97 RTF for Drawing Objects (Shapes)	Value
\shpfhdrN **	Word 97 RTF for Drawing Objects (Shapes)	Value
\shpgrp **	Word 97 RTF for Drawing Objects (Shapes)	Value
\shpleftN **	Word 97 RTF for Drawing Objects (Shapes)	Value
\shplidN **	Word 97 RTF for Drawing Objects (Shapes)	Value
\shplockanchor **	Word 97 RTF for Drawing Objects (Shapes)	Flag
\shppict **	Pictures	Destination

\shprightN **	Word 97 RTF for Drawing Objects (Shapes)	Value
\shprslt **	Word 97 RTF for Drawing Objects (Shapes)	Value
\shptopN **	Word 97 RTF for Drawing Objects (Shapes)	Value
\shptxt **	Word 97 RTF for Drawing Objects (Shapes)	Value
\shpwrkN **	Word 97 RTF for Drawing Objects (Shapes)	Value
\shpwrN **	Word 97 RTF for Drawing Objects (Shapes)	Value
\shpzN **	Word 97 RTF for Drawing Objects (Shapes)	Value
\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
\sprsbsp **	Document Formatting Properties	Flag
\sprslnsp *	Document Formatting Properties	Flag
\sprsspbf	Document Formatting Properties	Flag
\sprstsm **	Document Formatting Properties	Flag
\sprstsp	Document Formatting Properties	Flag
\staticval *	Information Group	Value
\stextflow **	Section Text	Value
\strike	Character Formatting Properties	Toggle
\strikedl **	Character Text	Toggle
\stylesheet	Style Sheet	Destination
\sub	Character Formatting Properties	Flag
\subdocument	Paragraph Formatting Properties	Value

\subfontbysize *	Document Formatting Properties	Flag
\subject	Information Group	Destination
\super	Character Formatting Properties	Flag
\swpbdr	Document Formatting Properties	Flag
\tab	Special Characters	Symbol
\tb	Tabs	Value
\tc	Table of Contents Entries	Destination
\tcelld **	New Control Words Created by Asian Versions of Word 97	Flag
\tcf	Table of Contents Entries	Value
\tcl	Table of Contents Entries	Value
\tcn	Table of Contents Entries	Flag
\template	Document Formatting Properties	Destination
\time **	Fields	Flag
\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	Document Formatting Properties	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
\truncatefont		
height *	Document Formatting Properties	Flag
\tx	Tabs	Value
\txe	Index Entries	Destination
\ucN **	Unicode RTF	Value
\ud **	Unicode RTF	Destination
\ul	Character Formatting Properties	Toggle
\uld	Character Formatting Properties	Flag
\uldash **	Character Text	Toggle
\uldashd **	Character Text	Toggle
\uldashdd **	Character Text	Toggle
\uldb	Character Formatting Properties	Flag
\ulnone	Character Formatting Properties	Flag
\ulth **	Character Text	Toggle
\ulw	Character Formatting Properties	Flag
\ulwave **	Character Text	Toggle
\uN **	Unicode RTF	Value
\up	Character Formatting Properties	Value
\upr **	Unicode RTF	Destination
\userprops *	Information Group	Destination
\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
\viewkindN **	Document Formatting Properties	Value
\viewscaleN **	Document Formatting Properties	Value
\viewzkN **	Document Formatting Properties	Value
\wbitmap	Pictures	Value
\wbmbitspixel	Pictures	Value
\wbmplanes	Pictures	Value
\wbmwidthbytes	Pictures	Value
\widctlpar	Paragraph Formatting Properties	Flag
\widowctrl	Document Formatting Properties	Flag
\windowcaption **	Document Formatting Properties	Value
\wmetafile	Pictures	Value
\wpeqn **	Fields	Flag
\wpjst **	Document Formatting Properties	Flag
\wpsp **	Document Formatting Properties	Flag
\wraptrsp	Document Formatting Properties	Flag
\xe	Index Entries	Destination
\xef	Index Entries	Value
\yr	Information Group	Value
\yxe **	New Control Words Created by Asian Versions of Word 97	Flag
\zwj	Special Characters	Symbol
\zwnj	Special Characters	Symbol

#######