WordPerfect
5.1, 5.2+ & 7C

WordPerfect 5.1 Document Format

This section describes the WordPerfect file format for WordPerfect 5.0 (WP5.0 DOS, UNIX, Data General, VAX), WordPerfect 5.1 (WP5.1 DOS, UNIX, VAX), WordPerfect 5.2+ and 7C (WP5.1 UNIX) and WordPerfect 5.1/5.2 for Windows (WPWin 5.1/5.2) documents.

Note: *WP5.1 (without an extension) refers collectively to WP5.1 DOS, WP5.1 VAX, WP5.1, 5.2+ and 7C UNIX, and WPWin 5.1/5.2.*

WordPerfect documents are binary files that contain both ASCII characters and special formatting codes. Each document has two major sections: a prefix area and a document area.

The prefix area contains:

- ►Information that identifies the file as a WP5.1/5.0 document
- ►Default values for margins, fonts, and other format settings
- ►Information about the selected printer
- ►Graphic images contained in the document

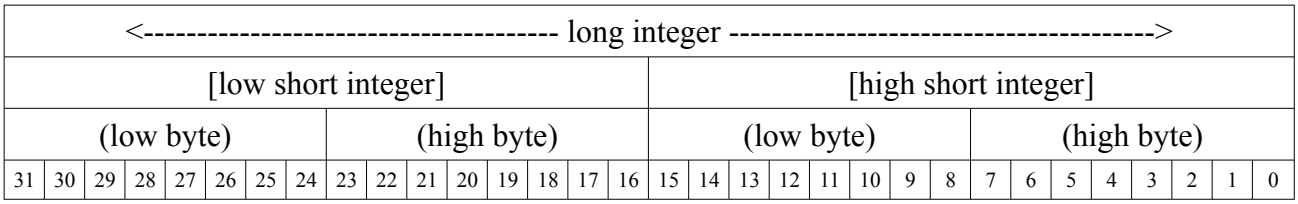
The document area contains the actual text and formatting codes that make up the bulk of the file.
Sizes are referred to as bytes, short (for short integers), or long (for long integers). Depending on the environment and operating system you are working on, these can be translated using the information in the following table.

Table WPF.1: Size Definitions

Byte	character (char)	byte	8 bits
Short	integer	word	16 bits
Long	long integer	double word	32 bits

All WordPerfect document files follow the Intel "byte reversed" method of storing short and long integer values. In this storage method the high byte of a short integer comes first, and the high short integer comes first in a long integer value. The following two block diagrams represent this storage format.

	<---------- short integer ---------->															
	[low byte]								[high byte]							
	(high nibble)				(low nibble)				(high nibble)				(low nibble)			
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0



Decimal numbers larger than 9 are followed, in parentheses "( )", by a number that begins with "0x". This represents the equivalent value in the *hexadecimal* number system. Single-digit decimal numbers stand alone, because such numbers have the same value in either hexadecimal or decimal. In most cases values are unsigned. An example of how unsigned values are documented is "255 (0xFF)." An example of how signed values are documented is "þ1 (0xFF)."

Measurements and positions can be stored in one of two units:

wpu = WordPerfect units (1 = 1/1200th of an inch, or 0.02117 mm)
su = Screen pitch units (1 = 1 screen display column)

The file format for the WP4.2 version contains only a document body with function codes. WP4.2 files do not have a prefix header. The WP42FF.TXT file on the Toolkit diskette (PC Developer's Toolkit only) contains the function codes for WP4.2.

Prefix Area

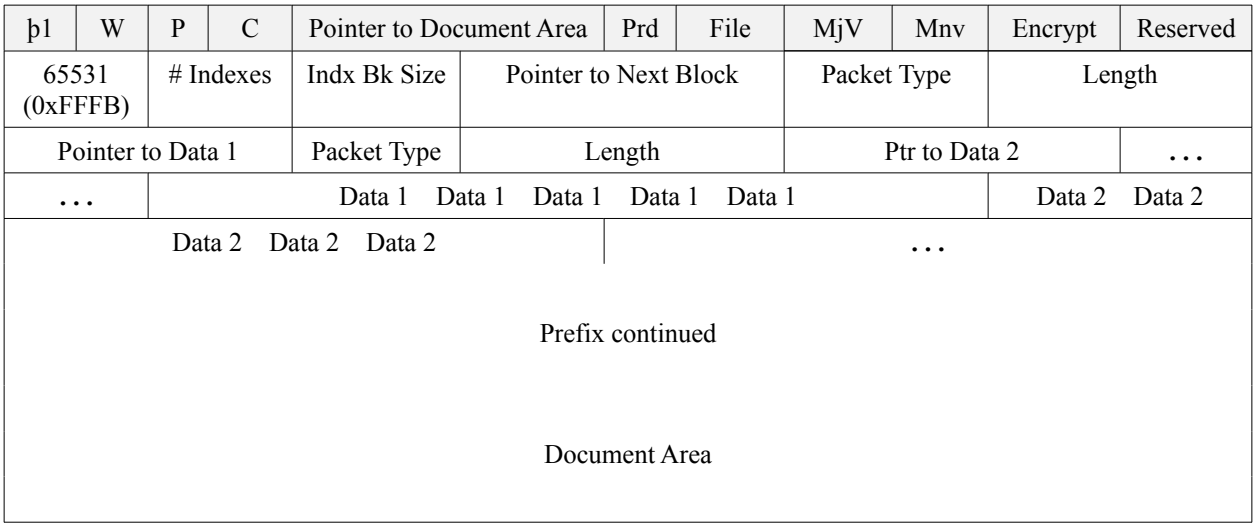
The prefix area of a WP5.1 document consists of a 16-byte file prefix followed by additional prefix information that is specific to the document. This can be any length, depending on the WordPerfect document and file prefix contents.

File Prefix

All files created by WPCorp products begin with a 16-byte file prefix that identifies the file type, product, and version. The file prefix has the following structure:

Field	Size	Value for WP5.1 Document
WPCorp File ID	4 bytes	þ1 (0xFF),"WPC"
Start of Document	long	File position of start of document
Product Type	1 byte	1
File Type	1 byte	10 (0xA)
Major Version #	1 byte	0
Minor Version #	1 byte	0 in WP5.0, 1 in WP5.1
Encryption Key	short	0 = not encrypted
Reserved	short	0

The following diagram illustrates the prefix area structure of a WP5.0, WP5.1, and WPWin 5.1 document. The shaded section of the diagram is the 16-byte file prefix that identifies a WPCorp file. The definitions of the 16-byte length are in byte, short, or long lengths.



WPCorp File ID

The WPCorp File ID fields are the first four bytes of a file and have the same values for all files produced by WPCorp products (excluding WP4.2). It is displayed as "þ1WPC" or "FF 57 50 43" in hexadecimal value. If you look at a WPCorp file in any binary program editor or ASCII editor, this ID ought to be there. If you do not see this ID, the file is not a WPCorp 5.x file.

Start of Document

The Start of Document field is a long value that begins at offset 4 in the 16-byte file prefix heading. This is a pointer to the beginning of the document area. Short integers and long integers are saved in a byte-reversed order.

Please note that if you place any codes such as margins, tabs, font changes, or a specific page size at the beginning of your document, those codes appear before the actual text begins.

Product Type Field

The Product Type field is one byte in length and is the ninth byte from the beginning of the file. It contains a value that identifies the WPCorp software product used to create the file. The field type can have one of the following values:

Value	Product
1	WordPerfect
2	Shell
3	Notebook
4	Calculator
5	File Manager
6	Calendar
7	Program Editor/Ed Editor
8	Macro Editor
9	PlanPerfect
10 (0xA)	DataPerfect
11 (0xB)	Mail
12 (0xC)	Printer (PTR.EXE)
13 (0xD)	Scheduler
14 (0xE)	WordPerfect Office
15 (0xF)	DrawPerfect
16 (0x10)	LetterPerfect

Please remember that other product type values may be added in the future.

File Type Field

The File Type field is one byte in length and is the tenth byte from the beginning of the file. The value depends on the Product Type (WPCorp software product) used to create the file. The first ten values (0–9) are reserved for *general purpose files* that have application across all WPCorp products. The values 10 and above are available for product-specific file types and are listed below in the WordPerfect File Types subsection.

General Purpose File Types

These general purpose file types are documented for your information only. These files types are specified for the use of WPCorp software.

Value	File
1	Macro file
2	Help file
3	Keyboard definition file
4	VAX keyboard definition file ***added for WP5.1 (3-30-90)***

Note: *The Help file does not have a prefix, but the 2 value is still reserved for the help file. Any third-party software or application can thus disregard the help file types.*

WordPerfect File Types

The file type for WPCorp products is one byte in length and identifies what type of file it is. The values are 16 (0x10) and above, and the file may not specifically be a document file.

The following file types are specific to WP5.0 and are also used for WP5.1. The following file types are specific to WPCorp:

Value	File
10 (0xA)	WordPerfect document
11 (0xB)	Dictionary file
12 (0xC)	Thesaurus file
13 (0xD)	Block
14 (0xE)	Rectangular block
15 (0xF)	Column block
16 (0x10)	Printer resource file (.PRS)
17 (0x11)	Setup file (contains the system values for WP{WP}.SET [Setup values, Shift-F1])
18 (0x12	Reserved
19 (0x13)	Printer resource file (.ALL)
20 (0x14)	Display resource file (.DRS)
21 (0x15)	Overlay file (WP.FIL)
22 (0x16)	WP graphic file (.WPG)
23 (0x17)	Hyphenation code module
24 (0x18)	Hyphenation data module
25 (0x19)	Macro resource file (.MRS)
26 (0x1A)	WP5.0 Graphics/Text Drivers
27 (0x1B)	Hyphenation lex module

The following file types were added to the WordPerfect file type for WP5.1:

Value	File
28 (0x1C)	Printer Q codes (used by VAX/DG)
29 (0x1D)	Spell code module—word list
30 (0x1E)	WP.QRS file (WP5.1 equation resource file)
31 (0x1F)	Reserved
32 (0x20)	VAX .SET
33 (0x21)	Spell code module—rules
34 (0x22)	Dictionary—rules

35 (0x23) Reserved
36 (0x24) WP5.1 Graphics/Text Drivers
37 (0x25) Rhymer word file (WPCorp product, TSR)
38 (0x26) Rhymer pronunciation file
39 – 40 (0x27 – 0x28) Reserved
41 (0x29) WP51.INS file (install options file)
42 (0x2A) Mouse driver for WP5.1
43 (0x2B) UNIX Setup file for WP5.0
44 (0x2C) MAC WP2.0 document
45 (0x2D) VAX file (WP4.2 document)
46 (0x2E) External Spell Code Module (WP5.1) (This file type is set aside for third parties to create their own speller code modules in WP5.1 DOS)
47 (0x2F) External Spell Dictionary (This file type is set aside for third parties to create their own dictionary ".LEX" files to be read by their speller code.)
48 (0x30) MAC SOFT graphics file (SOFT(Sequential Object FormaT) graphics file for the MAC WP)
49 - 50 (0x31 - 0x37) Reserved
51 (0x38) WPWin 5.1 Application Resource Library ***added for WPWin 5.1***

Major and Minor Version Field

The Major and Minor Version fields are each one byte in length. The value in the fields indicates the version number that the file type was created for. The value may or may not correspond to the product's published revision number. The Major version number for WordPerfect 5.x files is 0.

Note: *The "Minor Version #" for WP5.0 is 0. The "Minor Version #" was incremented for WordPerfect 5.1 (the value is 1). The "Minor Version #" for WPWin 5.1/5.2 is 1, the same number used for WP5.1.*

Encryption Key Field

The Encryption Key field is a short-integer value and occupies the 13th and 14th bytes. It must be a value 0 if the file is not encrypted; otherwise, it must contain a hash value of the password. This hash value is used as a check-sum to see if the user typed the correct password.

Reserved Field Type

Do not use the Reserved field. It must be zero (0).

Additional Prefix Area

The additional prefix area follows the 16-byte file prefix. Its structure depends on the product and file type. The additional prefix information for WP5.0 and WP5.1 documents is organized into index blocks and data packets. Each index block contains several indexes that point to data packets. The first index in each block is a special header index that gives the number of indexes in the block. It also gives the pointer to the next block. There can be more than one block (five indexes) in a prefix header, depending on how much information is stored in the prefix.

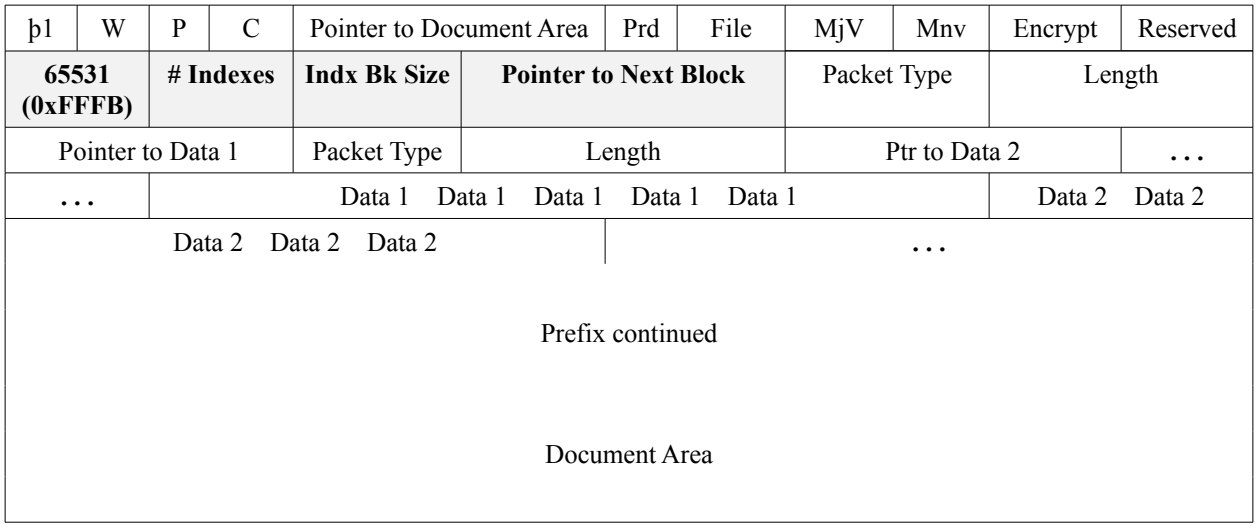
WP5.0 and WP5.1 currently store five, *and only five*, indexes in each index block (the special header index plus four general purpose indexes).

Each index block in a WP5.0/5.1 and WPWin 5.1/5.2 file consists of a special header index followed by four indexes that point to data packets. The data packets immediately follow the index block. The prefix area of the file consists of a sequence of index blocks and associated data chained together by the special header index in each index block.

Special Header Index

The *special header index* is 10 bytes long. It has the following structure (also illustrated in the shaded area of the diagram):

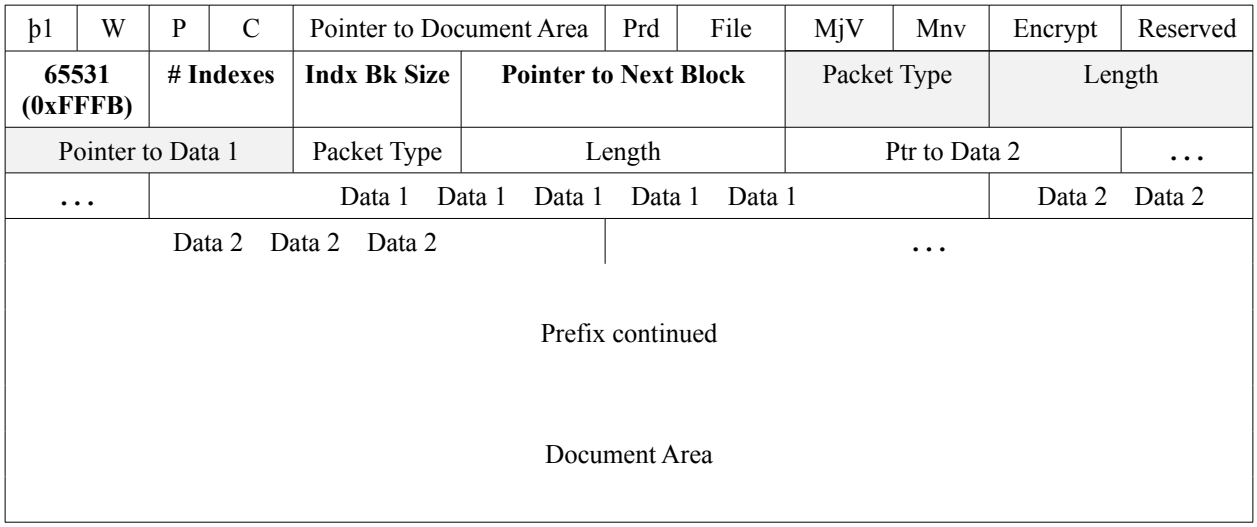
Offset	Size	Usage
0	short	Packet type for header index 65531 (0xFFFB)
2	short	# of indexes (including the special header, there are five in WP5.0 and WP5.1 documents.)
4	short	Size of index block 50 (0x32) (# of indexes * 10)
6	long	File position of next index block from beginning of file



General Packet Types

The *general packet types* indexes in the index block are also 10 bytes long. They have the following structure and are illustrated in the diagram below:

Offset	Size	Usage
0	short	Packet type
2	long	Length of data packet
6	long	File position of data for packet type



Special Data Packet Types

The following *special data packet types* are defined for use by all file types:

Value	Packet Type
0	End of prefix
65531 (0xFFFB)	Header index for index block
65532 (0xFFFC)	Reserved
65533 (0xFFFD)	Reserved
65534 (0xFFFE)	Reserved
65535 (0xFFFF)	Deleted packet (may still exist in file)

Notes: *The WP5.1 document prefix has the same structure as it does in WP5.0. However, some data packets have different values in WP5.1 and add new packet types for WP5.1 additional features. Note the following changes listed under packet types. WP5.0 and WP5.1 are forward and reverse compatible. When retrieved into WP5.0, additional features of WP5.1 are recognized as [UNKNOWN] codes. When the document is again retrieved into WP5.1, the [UNKNOWN] codes are recognized as their original features.*

> *If Fast Save is not active in WordPerfect 5.1, the file prefix is purged of deleted packets (65535 [0xFFFF]) when the file is saved. If Fast Save is active, deleted packets can still exist in the file prefix after the file is saved.*

WordPerfect Packet Types

The following are data packet types defined for WP5.0/5.1, and WPWin 5.1/5.2 documents that are found in the additional prefix area. They are used or referred to in the *general packet types* structure.

Value	Packet Type
1	Document summary packet
2	List of fonts used in document (WP5.0)
3	Document initial codes
4	Reserved
5	Reserved
6	Document specific flags
7	Font name string pool
8	Graphics information
9	Form hash table
10 (0xA)	Reserved
11 (0xB)	Reserved
12 (0xC)	Document printer (.PRS file name, etc.)
13 (0xD)	Reserved
14 (0xE)	Supplemental dictionary compressed words
15 (0xF)	List of fonts used in document (WP5.1)
16 (0x10)	DDE link packet added for WPWin 5.1 (In UNIX 5.1, 5.2+ and 7C only, this packet # indicates the # of pages)
17 (0x11)	Macro executable code added for WPWin 5.1
18 (0x12)	Reserved
19 (0x13)	Macro information block added for WPWin 5.1
20 (0x14)	Reserved
21 (0x15)	OLE prefix added for WPWIN 5.2
256 (0x100)	First style packet # (use 256 packets for styles)
511 (0x1FF)	Last style packet # (use 256 packets for styles)
512 (0x200)	PS table packet — first packet
767 (0x2FF)	PS table packet — last packet

WP Packet Type Structure for Packets 1 - 767 (1 - 0x2FF)

The following are the related structures to the *WordPerfect document prefix packet types:*

Packet Type 1 — Document Summary Packet

Except for the fields marked and the last 3 fields in the WP5.1 format, all fields are null terminated strings.

If a user imports a 5.0 document file into the 11/6/89 release of WP5.1, and the file had a

summary packet in it, the following could happen: If the user never edited the summary information, this version of 5.1 would not rewrite the 5.0 summary packet. A document saved out of this version of 5.1, labeled as being a 5.1 document, but has a 5.0 formatted summary packet. Keep this in mind if your program is reading WP5.1 files.

Field	WP5.1 Max Len	WP5.0 Max Len
Creation date (string form)	† 26 (w/null)	26 (w/null)
Descriptive name	† 68 (no null)	41 (w/null)
Descriptive type	† 21 (w/null)	N/A
Subject	161	40
Author	61	40

† *Lengths are padded with spaces to be exact, so that a document management program can change the name or type without completely rewriting the prefix packet.*

Typist	61	40
Abstract (comments in WP5.0)	781	780
End of WP5.0 Document Summary		
Account	161	N/A
Key words	161	N/A
WP5.1 Format Marker	1	N/A (always equals -1 [Ffh])
Creation date (10-byte format)	10	N/A
WP5.1 Format Marker	1	N/A (always equals -1 [Ffh])

Note: *"WP5.1 Format Markers" are used by WP to determine whether the summary packet is in the format used by WP5.0 (the markers don't exist) or the expanded format of WP5.1. If the end of the packet data has the value 0FFh (-1), then it is a WP5.1 format. If the packet ends with zero, then it is a WP5.0 packet.*

Packet Type 1 and the 10-byte Creation Date format for the Document Summary Packet is as follows:

Offset	Size	Meaning
0	short	Year
2	1 byte	Month
3	1 byte	Day
4	1 byte	Hour
5	1 byte	Minute

The next four bytes are unused (0) in the Document Summary Packet, but are:

6	1 byte	Second
7	1 byte	Day_of_week
8	1 byte	Time_zone
9	1 byte	Unused

Packet Type 2 — List of Fonts Used in Document (WP5.0)

This packet is a variable-length list of records. To determine the number of these records, divide the size of the packet by the size of the record structure below. The structure of these records is

as follows:

Offset	Size	Usage
0	short	Font ID (instance pointer)
2	short	Cache ID of PS table (0xFF [þ1] = no PS table)
4	short	Top shoulder height (psu)
6	short	Bottom shoulder height (psu)
8	short	Printed char width (wpu)
10	short	Desired space width (wpu)
12	short	Normal space width (wpu)
14	short	Horizontal motion units
16	short	Numerator for fractional units of horizontal motion
18	short	Pointer into font string pool of font names
20	1 byte	Flags
21	1 byte	Available (to ensure short integer alignment)
22	short	Printer point size in 3600ths (same as offset 26 if not scalable)
24	short	Hash
26	24 bytes	Offsets 0–20 of Font Descriptor (See *Packet 15 [0xF]* for format)
50	16 bytes	AFC (Automatic Font Changes) list (attribute list)
66	16 bytes	Character AFC list
82	4 bytes	Orientation AFC list

Packet Type 3 — Document Initial Codes
This variable-length packet contains function codes as they appear inside a document. *Setup Initial Codes* inserts the function codes originally, but you can modify them separately. They keep the system's setup environment and the document together when both are moved to a different environment.

Packet Type 4 — Reserved (not used)

Packet Type 5 — Reserved (not used)

Packet Type 6 — Document-Specific Flags and Information
This packet contains information about the document itself at the time it was saved. The format of Packet Type 6 is as follows:

Offset	Size	Usage
0	1 byte	Flag bits for document bit 0: 0 = WPWIN Wysiwyg manual display pitch 1 = WPWIN Wysiwyg auto display pitch bits 1 & 2: 0 = font change 1 = red color 2 = char in left margin 3 = char in alternating margins bit 3 = 1 if document must be formatted bit 4 = 1 if document must be (re)generated bit 5 = 1 if manual display pitch bit 6 = 1 if links must be updated on retrieval bit 7 = 1 if links codes are not to be displayed
1	1 byte	Low nibble = print quality High nibble = graphics quality For each nibble: 0 = don't print 1 = draft 2 = med 3 = high
2	short	Redline character
4	short	Width of screen character
6	short	Binding width of document
8	1 byte	Printer select cnt flag for eqn hash
9	1 byte	Reserved
10	short	Document display pitch. 0x100 = 100%
12	short	Screen resolution
14	short	Reserved

Packet Type 7 — Font Name String Pool

This variable-length packet contains the font names in the font list. The font list records each have a field that points to an offset in this packet that indicates its font name. These are null-terminated, font-name strings.

Packet Type 8 — Graphics Information

Give special handling to the graphics image data (Packet Type 8). The index for this data packet must be the last index in the prefix area (except for an End of Prefix index). The value of the long integer length given in the index is 2. The graphic image data has the following structure:

Offset	Size	Usage
0	short	Count of # of graphic images
2	long	Size of first graphic image
6	long	Size of second graphic image
10	long	Size of third graphic image
..	..	..
..	..	..
(# graphics * 4)+2	variable	Data for first graphic image
+ size 1	variable	Data for second graphic image
+ size 2	variable	Data for third graphic image
..	..	..
..	..	..
..	..	..

Packet Type 9 — Forms Hash Table
This variable-length packet is a list of hash values (short integers). The hash values correspond to each form that the user defined. To determine the number of forms hashed, divide the packet size by 2.

Packet Type 10 (0xA) — Reserved

Packet Type 11 (0xB) — Reserved

Packet Type 12 (0xC) — Document Printer (.PRS file, etc.)
This fixed-size packet contains information about the selected printer. The structure is as follows:

Offset	Size	Description
0	37 bytes	Long name of .PRS file
37	13 bytes	Actual .PRS filename
50	24 bytes	Descriptor for initial font
74	short	Minimum top margin
76	short	Minimum bottom margin
78	short	Minimum left margin
80	short	Minimum right margin
82	1 byte	Flags bit 0 = used by printer selection to mark selected printer bit 1 = initialize printer when WP starts bit 3 = right hzone disabled bits 4,5,6,7 = orientations supported
83	short	Date of .PRS file bits 0–4 = day (starting at 1) bits 5–8 = month bits 9–15 = year+80

Offset	Size	Description
85	short	Time of .PRS file bits 0–4 = seconds/2 (may not be too accurate) bits 5–10 = minutes bits 11–15 = hour (24 hour starting at 0)
87	short	Point size of current font
89	1 byte	Typeface flags

Packet Type 13 (0xD) — Reserved

Packet Type 14 (0xE) — Supplemental Dictionary Compressed Words

Packet Type 15 (0xF) — List of Fonts Used in Document (WP5.1)
Structure of Prefix Font Information
The following 86-byte structure is repeated for each font in a document:

Offset	Size	Description	Where in .PRS File
0	short	Font ID (instance pointer)	Font def
2	short	Cache ID of PS table (þ1 [0xFF] = no PS table)	Font def
4	short	Top shoulder height (psu)	
6	short	Bottom shoulder height (psu)	
8	short	Printed char width (wpu)	
10	short	Optimal space width (wpu)	
12	short	Normal space width (wpu)	
14	short	Horizontal motion units	
16	short	Numerator for fraction units of horizontal motion	Font def
18	short	Pointer in font string pool of font name (see *Packet 7*)	Font def
20	1 byte	Flags	
21	short	Hash of descriptor (“Match font hash value” in Function 209 (0xD1) Subfunction 1)	
23	27 bytes	Font descriptor (see *Font Descriptor* below)	
50	16 bytes	AFC list (prffont entry #'s, þ1 (0xFF) = none)	
66	16 bytes	Char afc list	
82	4 bytes	Orientation afc list	

Font Descriptor

Offset	Size	Description	Where in .PRS File
0	short	Cell height (5.0)(3600ths)	Font def
2	short	Optimal width (wpu)	Font def
4	short	Cap height (psu)	Typeface def

Offset	Size	Description	Where in .PRS File
6	short	x height (psu)	Typeface def
8	short	Descender height (psu)	Typeface def
10	short	Italic adjustment (psu, ±)	Typeface def
12	3 bytes	Typeface descriptor	Typeface def
		Low short integer: bit 0 = 1 casual bit 1 = 1 connected letters bit 2 = 1 decorative bit 3 = 1 formal bit 4 = 1 future bit 5 = 1 old style bit 6 = 1 script or calligraphic bit 7 = 1 cupped bit 8 = 1 exaggerated bit 9 = 1 hairline bit 10 = 1 slab bit 11 = 1 slanted bit 12 = 1 transitional bit 13 = 1 triangular bit 14 = 1 curved or bowed diagonals bit 15 = 1 nonconnecting enclosures	
		High byte: bit 0 = 1 round bit 1 = 1 square bit 2 = 1 angular bit 3 = 1 exaggerated bit 4 = 1 uniform bit 5 = 1 ball terminals bit 6 = 1 tails bit 7 = 1 reserved; must = 0	
15	1 byte	Typeface def flags; same format as offset 26, excluding the information added for WP5.1	Typeface def
(for WP5.0)			
16	1 byte	Hash of typeface name	Typeface def
17	1 byte	Quality of the font (only relevant for .DRS fonts)	Font def
18	short	Hash of font name	Font def
20	2 short int	Charset completeness bits (The completeness bits are read from each of the 2 short integers.)	Non-shareable font Def size

Low Short Int.	High Short Int.	Meaning
Bit n =	Bit n =	
0	0	Charset not supported
1	0	Charset supported 1%–25%
0	1	Charset supported 26%–59%
1	1	Charset supported 60%–100%

Note: *The "n" in "Bit n =" equals the number of the character set. For example, if you wanted to check the completeness of character set zero (the ASCII character set), you would look at bit zero (0) of the low and high short integers.*

Offset	Size	Description	Where in .PRS File
24	short	Point size (5.1) (3600ths)	Font def
26	1 byte	Typeface def flags	Typeface def
		1 Small caps font	
		2 Shadow font	
		4 Outline font	
		8 Italic or oblique	
		16 (0x10)	PS font (font descriptor only)
		Weight — high three bits	
		00000000b extra light	
		00100000b light	
		01000000b normal	
		01100000b demi-bold	
		10000000b bold	
		10100000b heavy *added for WP5.1*	
		11000000b black *added for WP5.1*	
		11100000b ultra-black *added for WP5.1*	

Packet Type 16 (0x10) — DDE Link Packet *added for WPWin 5.1*
The existence of this packet signifies that the document contains DDE links. The presence of this packet tells the Open and Retrieve functions to scan the document and reestablish the links.

Packet Type 16 (0x10) — Total Number of Document Pages *UNIX WP 5.1, 5.2+, 7C only*
0 long # of pages

Packet Type 17 (0x11) — Macro Executable Code *added for WPWin 5.1*
This packet is a part of every macro file.

Packet Type 18 (0x12) — Reserved

Packet Type 19 (0x13) — Macro Information Block *added for WPWin 5.1*

Offset	Size	Description

0 10 bytes Creation date (if this is altered, the macro is recompiled at run-time)

Note: *For creation date format, see* Document Summary Packet (prefix packet 1).

10	short	File type
12	short	Major version
14	short	Minor version
16	short	Maintenance version
18	short	Language (for example, US, UK, etc.)
19	short	# of product types in macro

The following section is repeated for each product type in offset 19 (above):

..	2 short int.	Product (for example, WPWP)
..	short	Product major version
..	short	Product minor version
..	short	Product maintenance version

Packet Type 20 (0x14) — Reserved

Packet Type 21 (0x15) — OLE Prefix *added for WPWin 5.2*

Offset	Size	Description
0	1 byte	Object Information Structure major version (currently always 1)
1	1 byte	Object Information Structure minor version (currently always 0)
2	short	Size of Object Information Structure (currently always 4)
4	short	# of OLE objects

Note: *The following information repeats for each object in the packet. Offset 4 determines how many times the information will be repeated.*

6	short	Size of object name (includes the NULL)
8	variable	Object name (null-terminated ANSI string)
..	variable	Object information structure (currently contains only an unsigned long integer value. The value is the size of the OLE object)
..	variable	OLE object

Packet Type - Styles 256-511 (0x100 - 0x1FF)

Prefix structure of style packet types. The style library file format is the same as the style prefix information.

A style is stored in the prefix in two parts:

PART_1:	Name	Type	Description	Unique#			
sizes:	21	1	55	1			
PART_2:	BegSize	BegHash	EndSize	EndHash	XtrSize	BegText	EndText
sizes:	2	2	2	2	2	**..**	**..**

Field	**Meaning**
Name	A short name for the style, the actual identifier for the style
Type	Information about the style bit 0 set = paired type style bit 0 clear = global type style bit 6,7 0 = Enter = HRt 2 = Enter = Off 3 = Enter = Off/On
Description	A lengthy description for human use only
Unique #	A # to identify this style from others
BegSize	The size of the beginning text
BegHash	The hash value of the beginning text
EndSize	The size of the ending text
EndHash	The hash value of the ending text
XtrSize	The size of the extra text (auto-added to End_Text)
BegText	Text and function codes in the Style On code
EndText	Text and function codes in the Style Off code
Xtrtext	Codes added to the Style Off code

Packet Type 512 - 767 (0x200 – 0x2FF)
Structure of Prefix PS Information
The first three fields uniquely identify a PS table entry:

Offset	**Size**	**Meaning**
0	1 byte	PS status flag (0 = bytes, 1 = short int.)
1	1 byte	Reserved
2	16 short int.	Pointers to the 16-character sets (þ1 = pointer is being used)
34	variable	Char widths
..	**..**	._____.
..	char set 1 þ	| Last char or count | **Note:** *last char + 1 =* ***count***
..	**..**	| char 0 width |
..	**..**	| char 1 width |
..	**..**	| **..** |
..	**..**	| **..** |
..	**..**	| **..** |

Offset	Size	Meaning
..	..	._____.
..	char set 2 þ	| Last char or count |
..	..	| char 0 width |
..	..	| char 1 width |
..	..	| .. |
..	..	| .. |

Abbreviated Prefix
One of the following two conditions signals the end of the prefix data: when an index with a packet type of End of Prefix (0) is encountered, or when the pointer to the next index block in the special header index is equal to 0.

To quickly create a simple but valid WordPerfect document prefix, use the following information. This abbreviated prefix enables successful retrieval into WordPerfect. WordPerfect adds the remaining information when the document is saved after bringing it into WordPerfect (The following values are all in hexadecimal).

```
FF  57  50  43  4C  00  00  00    01  0A  00  00  00  00  00  00
FB  FF  05  00  32  00  00  00    00  00  06  00  08  00  00  00
42  00  00  00  08  00  02  00    00  00  4A  00  00  00  00  00
00  00  00  00  00  00  00  00    00  00  00  00  00  00  00  00
00  00  08  00  7C  00  78  00    00  00  00  00
```

Note: *This generic prefix is on disk so you do not have to type it in. It is saved under the filename "PREFIX.QCK." Refer to the* Toolkit Diskette *section to find which disk this file is on.*

Document Area

This section contains information on the document area of a WordPerfect 5.1 file that uses single-byte and multi-byte codes.

Single-byte codes include control characters (values 1 to 31 [0x1F]), ASCII characters (values 32 to 126 [0x20 to 0x7E], space to tilde), and single-byte functions (values 128 to 191 [0x80 to 0xBF]). The values 0, 255 (0xFF), and 127 (0x7F) must not appear as single codes in a WordPerfect document.

Multi-byte codes include fixed- and variable-length functions. These functions consist of a group of bytes rather than a single byte and always begin and end with a function code (values 192 to 255 [0xC0 to 0xFF]) that indicates the type of function.

The multi-byte function codes are separated into two groups: fixed-length and variable-length. The fixed-length functions have a predetermined length and have the following structure:

Begin function code (192–207 [0xC0–0xCF])
Data bytes
..
..
End function code (same as begin function code)

The smallest fixed-length, multi-byte function is at least three bytes long and consists of the function code, a single data byte, and the function code again.

Each variable-length, multi-byte function has a subfunction code that follows the function code and a short integer (16 bits), Length on each end of the function:

Begin function code (byte) (208–255 [0xD0–0xFF])
Begin subfunction code (byte)
Length (short)
Data bytes
..
..
Length (short)
End subfunction code (byte) (same as begin subfunction code)
End function code (byte) (same as begin function code)

Note: *When creating a multi-byte function, the "old values" should be reserved with a 0 (null). WordPerfect reformats retrieved files, inserts old values along with the existing new values, and formats according to the new values (with the exception of footnotes). WordPerfect compares the old values with the new values. WordPerfect formats to the set defaults — or as close to the original as WP5.1 can calculate.*

The hash value is an internal function of WordPerfect that does not have a purpose for a third-

party interface. A value of 0 can be placed in the code for any hash value to reserve space in the function. This lets WordPerfect use that offset internally, as needed.

The smallest variable-length, multi-byte function is at least nine bytes long. The function code for a variable-length function typically indicates a "class" or group of functions, and the subfunction code gives the specific function within that group. The Length field is always the number of data bytes in the function minus four. (The length does not include the beginning Length and function codes.)

Text Fields

A variable-length, multi-byte function code that has a text field in it (that is, "Footnote text," "Endnote text," "Text box text," etc.) obeys all of the same rules as the body of the WordPerfect document area.

This text field can have many of the same function codes that are found in the body of the document. So don't assume that there is only text in these fields any more than you would assume that there is just text in the main body of a WordPerfect document file. This "nesting" of function codes inside of the text fields of other function codes is quite common and should always be expected.

The following sections contain lists of the five different code types found in a WordPerfect 5.1 document: control characters; ASCII characters; single-byte functions; fixed-length, multi-byte functions; and variablelength, multi-byte functions.

Note: *When you design code to read a WordPerfect document, remember that WPCorp codes may be added to at any time. So design your reading code to encounter any possible code, and not just the specific, unreserved codes that are documented here.*

If your reading code doesn't recognize the code, it should skip over the code and read the next piece of information in the file.

As a help, the defined lengths of the unused 200–207 (0xC8–0xCF) function codes are documented at the end of the Fixed-Length, Multi-Byte Functions *section.*

Control Characters (0x01 to 0x1F)

Value	Control Character	Meaning
1	Ctrl-A	Reserved
2	Ctrl-B	Page # printed here
3	Ctrl-C	† Merge codes C
4	Ctrl-D	† Merge codes D
5	Ctrl-E	† Merge codes E
6	Ctrl-F	† Merge codes F
7	Ctrl-G	† Merge codes G
8	Ctrl-H	Reserved
9	Ctrl-I	Reserved
10 (0xA)	Ctrl-J	Hard return
11 (0xB)	Ctrl-K	Soft page
12 (0xC)	Ctrl-L	Hard page
13 (0xD)	Ctrl-M	Soft return
14 (0xE)	Ctrl-N	† Merge codes N
15 (0xF)	Ctrl-O	† Merge codes O
16 (0x10)	Ctrl-P	† Merge codes P
17 (0x11)	Ctrl-Q	† Merge codes Q
18 (0x12)	Ctrl-R	† Merge codes R
19 (0x13)	Ctrl-S	† Merge codes S
20 (0x14)	Ctrl-T	† Merge codes T
21 (0x15)	Ctrl-U	† Merge codes U
22 (0x16)	Ctrl-V	† Merge codes V
23 (0x17)	Ctrl-W	Reserved
24 (0x18)	Ctrl-X	Reserved
25 (0x19)	Ctrl-Y	Reserved
26 (0x1A)	Ctrl-Z	Reserved
27 (0x1B)	Ctrl-[	Reserved
28 (0x1C)	Ctrl-\	Reserved
29 (0x1D)	Ctrl-]	Reserved
30 (0x1E)	Ctrl-^	Reserved
31 (0x1F)	Ctrl-_	Reserved

† See the *WordPerfect 5.0 Reference Manual*.

For information about the DOS third-party interface codes, see the *DOS Third-Party Interface* section of the *PC Developer's Toolkit* manual.

ASCII Characters (32 to 126 [0x20 to 0x7E])

Value	Character	Value	Character
32 (0x20)	Space	80 (0x50)	P
33 (0x21)	!	81 (0x51)	Q
34 (0x22)	"	82 (0x52)	R
35 (0x23)	#	83 (0x53)	S
36 (0x24)	$	84 (0x54)	T
37 (0x25)	%	85 (0x55)	U
38 (0x26)	&	86 (0x56)	V
39 (0x27)	'	87 (0x57)	W
40 (0x28)	(	88 (0x58)	X
41 (0x29)	)	89 (0x59)	Y
42 (0x2A)	*	90 (0x5A)	Z
43 (0x2B)	+	91 (0x5B)	[
44 (0x2C)	,	92 (0x5C)	\
45 (0x2D)	-	93 (0x5D)	]
46 (0x2E)	.	94 (0x5E)	^
47 (0x2F)	/	95 (0x5F)	_
48 (0x30)	0	96 (0x60)	`
49 (0x31)	1	97 (0x61)	a
50 (0x32)	2	98 (0x62)	b
51 (0x33)	3	99 (0x63)	c
52 (0x34)	4	100 (0x64)	d
53 (0x35)	5	101 (0x65)	e
54 (0x36)	6	102 (0x66)	f
55 (0x37)	7	103 (0x67)	g
56 (0x38)	8	104 (0x68)	h
57 (0x39)	9	105 (0x69)	i
58 (0x3A)	:	106 (0x6A)	j
59 (0x3B)	;	107 (0x6B)	k
60 (0x3C)	<	108 (0x6C)	l
61 (0x3D)	=	109 (0x6D)	m
62 (0x3E)	>	110 (0x6E)	n
63 (0x3F)	?	111 (0x6F)	o
64 (0x40)	@	112 (0x70)	p
65 (0x41)	A	113 (0x71)	q
66 (0x42)	B	114 (0x72)	r
67 (0x43)	C	115 (0x73)	s
68 (0x44)	D	116 (0x74)	t
69 (0x45)	E	117 (0x75)	u
70 (0x46)	F	118 (0x76)	v
71 (0x47)	G	119 (0x77)	w
72 (0x48)	H	120 (0x78)	x
73 (0x49)	I	121 (0x79)	y
74 (0x4A)	J	122 (0x7A)	z
75 (0x4B)	K	123 (0x7B)	{
76 (0x4C)	L	124 (0x7C)	|
77 (0x4D)	M	125 (0x7D)	}
78 (0x4E)	N	126 (0x7E)	~
79 (0x4F)	O		

Single-Byte Functions (128 to 191 [0x80 to 0xBF])

Value	Meaning
128 (0x80)	Temporary (always deleted)
129 (0x81)	Right justification ON **(WP5.0)**
130 (0x82)	Right justification OFF **(WP5.0)**
131 (0x83)	Soft end center/align
132 (0x84)	Reserved
133 (0x85)	Temporary (place saver)
134 (0x86)	Center page top to bottom
135 (0x87)	Columns ON
136 (0x88)	Columns OFF (at top of page)
137 (0x89)	Reserved
138 (0x8A)	Widow/Orphan ON
139 (0x8B)	Widow/Orphan OFF
140 (0x8C)	Combination hard return/soft page *added for WP5.1*
141 (0x8D)	Footnote/Endnote # (inside footnote/endnote)
142 (0x8E)	Figure # (inside caption)
143 (0x8F)	Hard end of center/align *added for WP5.1*
144 (0x90)	Deletable return at end of line (DSRt)

Note: *The WP formatter creates the DSRt. It cannot be used by programmers.*

145 (0x91)	Deletable return at end of page (DSPg)
146 (0x92)	End of page that is deleted when forward formatted
147 (0x93)	Invisible return in line (SRt that turns into a space)

Note: *This is the **Hard Space** (Home, Space Bar) function command. See* WordPerfect Reference.

148 (0x94)	Invisible return at end of line (147 [0x93] turns into this code at end of line)
149 (0x95)	Invisible return at end of page (147 [0x93] turns into this code at end of page)
150 (0x96)	Block ON
151 (0x97)	Block OFF
152 (0x98)	Table of contents page # place holder
153 (0x99)	Dormant hard return *added for WP5.1*
154 (0x9A)	Cancel hyphenation
155 (0x9B)	End of generated text (end of table of contents)
156 (0x9C)	Reserved
157 (0x9D)	Reserved
158 (0x9E)	Hyphenation OFF
159 (0x9F)	Hyphenation ON
Value	**Meaning**
160 (0xA0)	Hard space
161 (0xA1)	Do subtotal

162 (0xA2)	Subtotal entry
163 (0xA3)	Do total
164 (0xA4)	Total entry
165 (0xA5)	Do grand total
166 (0xA6)	Calculation column
167 (0xA7)	Math ON
168 (0xA8)	Math OFF
169 (0xA9)	Hard hyphen in line (The **<Home><->** command in the user interface)
170 (0xAA)	Hard hyphen at end of line (169 [0xA9] is converted to this when at end of line)
171 (0xAB)	Hard hyphen at end of page (169 [0xA9] is converted to this when at end of page)

Note: *Codes 170 (0xAA) and 171 (0xAB) may not be at the end of a line or page if the document was saved with "Fast Save" set to "Yes."*

172 (0xAC)	Soft hyphen in line (The <CTRL><-> command from the user interface)
173 (0xAD)	Soft hyphen at end of line (172 [0xAC] is converted to this code if at end of line)
174 (0xAE)	Soft hyphen at end of page (172 [0xAC] is converted to this code if at end of page)
175 (0xAF)	Columns OFF at end of line
176 (0xB0)	Columns OFF at end of page
177 (0xB1)	Math negate
178 (0xB2)	Outline OFF *added for WP5.1*
179 (0xB3)	Reverse Direction On *Hebrew/Arabic versions only*
180 (0xB4)	Reverse Direction Off *Hebrew/Arabic versions only*
181-188 (0xB5–0xBC)	Reserved
189 (0xBD)	Auto code placement on *added for WP5.1, 5.2+ & 7C Unix only*
190 (0xBE)	Auto code placement off *added for WP5.1, 5.2+ & 7C Unix only*
191 (0xBF)	Unknown single-byte function

Fixed-Length, Multi-Byte Functions (192 to 207 [0xC0 to 0xCF])

Function Code: 192 (0xC0)
Function Name: Extended Character
Size in Bytes: 4

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function code (192 [0xC0])
	1	1 byte	Character code
	2	1 byte	Character set (0–0–12 [0xC]) See *WordPerfect Reference*.
	3	1 byte	End function code (192 [0xC0])

Function Code: 193 (0xC1)
Function Name: Center/Align/ Tab/Left Margin Release
Size in Bytes: 9

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function code (193 [0xC1])
	1	1 byte	**Flags:**
			bit 0 = reserved
			bit 1 = 0 if the type (tab, align, etc.) is hard

added for WP5.1

= 1 if soft tab

Note: *Soft tabs automatically change according to the tab stop settings. For example, if the soft tab code is defined as a left tab and rests on a tab stop that is defined as a right tab setting, then the tab code is converted to a [Rgt Tab] code. Hard tab codes do not change, regardless of the tab stop setting.*

bit 2 = 1 this bit will be set when a tab is entered within a table, for positioning within the table cell (Ctrl-V, Tab) *added for WP5.1*

bit 3 = 0 align on alignment character or center on column

= 1 right-justified tab or centered tab

bit 4 = 1 dot leader

bit 5 = 1 center between margins, flush right to right margin

bits 6–7

= 0 tab
= 1 align (bit 6 set if center or align)
= 2 left margin release
= 3 center

	2	short	Absolute position of start text (wpu)

	Offset	Size	Meaning
	4	short	Absolute center/align/tab position (wpu)
	6	short	Screen column position of start text (su) *modified for WPWin 5.1*

Note: *WordPerfect automatically calculates the previous three values if set to 0.*

	8	1 byte	End function code (193 [0xC1])

Note: *If the 193 (0xC1) function code is used as a Tab or Margin Release, it is an "open" code with no "off" code. If this function code is used as a Center or Align code, it is a paired code and has one of two possible, single-byte function codes to end it. These two codes are 131 (0x83) or 143 (0x8F). See the* Single-Byte Functions *section for information on these two ending codes.*

Use the following formula to determine if a document came from DOS or Windows. Offset 2 is position in wpu's; Offset 6 is position in screen units.

```
if( (screen units > 0) && (document is WP5.1 format) )
then
  result = position wpu's/screen units
  if ( (result > 3) && (result < 30) )
        then document is a Windows type
                column to view document = position wpu's/120

  else document is DOS type
                column to view document = screen units

  end if
else document is WP5.0 format
                column to view document = screen units
end if
```

Function Code: 194 (0xC2)
Function Name: Indent
Size in Bytes: 11

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function code (194 [0xC2])
	1	1 byte	**Flags:** bit 0 = 0 left indent = 1 left /right indent bit 4 = 1 dot leader
	2	short	Difference between old and new temp margins (wpu)
	4	short	Absolute position of start of text (wpu) *("Old Current Column Number [su]" in WP5.0)*
	6	short	Absolute indent position (wpu)

	Offset	Size	Meaning
	8	short	Screen column position of start of text (su) *modified for WPWin 5.1*
	10	1 byte	End function code (194 [0xC2])

Function Code: 195 (0xC3)
Function Name: Attribute ON
Size in Bytes: 3

Structure:	Offset	Size	Meaning
	0	1 byte	Begin function code (195 [0xC3])
	1	1 byte	Attribute type 0 = extra large 1 = very large 2 = large 3 = small 4 = fine 5 = superscript 6 = subscript 7 = outline 8 = italics 9 = shadow 10 (0xA) = redline 11 (0xB) = double underline 12 (0xC) = bold 13 (0xD) = strikeout 14 (0xE) = underline 15 (0xF) = small caps
	2	1 byte	End function code (195 [0xC3])

Function Code: 196 (0xC4)
Function Name: Attribute OFF
Size in Bytes: 3

Structure:	Offset	Size	Meaning
	0	1 byte	Begin function code (196 [0xC4])
	1	1 byte	Attribute type (same as Attribute ON)
	2	1 byte	End function code (196 [0xC4])

Function Code: 197 (0xC5)
Function Name: Block Protect
Size in Bytes: 5

Structure:	Offset	Size	Meaning
	0	1 byte	Begin function code (197 [0xC5])

1	1 byte	Definition byte

	Offset	Size	Meaning
			0 = Block Protect ON
			1 = Block Protect OFF, outside of columns and inside of columns defined as "Parallel with Block Protect ON"
			3 = Block Protect OFF inside any other type of column.
	2	short	# of vertical 1200ths in block (wpu)
	4	1 byte	End function code (197 [0xC5])

Function Code: 198 (0xC6)
Function Name: End of Indent
Size in Bytes: 6

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function code (198 [0xC6])
	1	short	effective temporary right margin for the indent code associated with this end of indent (wpu)
	3	short	effective temporary left margin for the indent code associated with this end of indent (wpu)
	5	1 byte	End function code (198 [0xC6])

Function Code: 199 (0xC7)
Function Name: Different Display Character when Hyphenated (applies only to International Language Modules)
Size in Bytes: 7

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function code (199 [0xC7])
	1	1 byte	**Flags:**
			bit 0 = 0 if no hyphenation next to function
			= 1 if word is hyphenated next to function
			bit 1 = 0 if function is after the hyphenation
			= 1 if function precedes the hyphenation
	2	short	Character when in line
	4	short	Character when hyphenated
	6	1 byte	End function code (199 [0xC7])

Note: *If character is a null, nothing is displayed.*

Note: *Function codes 200 through 207 (0xC8 through 0xCF) are reserved for future use. They have the following defined lengths, so you can make your program upwardly compatible.*

Function Code: 200 (0xC8)
Size in Bytes: 4

Function Code: 201 (0xC9)
Size in Bytes: 5

Function Code: 202 (0xCA)
Size in Bytes: 6

Function Code: 203 (0xCB)
Size in Bytes: 6

Function Code: 204 (0xCC)
Size in Bytes: 8

Function Code: 205 (0xCD)
Size in Bytes: 10

Function Code: 206 (0xCE)
Size in Bytes: 10

Function Code: 207 (0xCF)
Size in Bytes: 12

Variable-Length, Multi-Byte Functions (208 to 255 [0xD0 to 0xFF])

The following is a summary of the variable-length, multi-byte functions.

Unknown Function Group 254 (0xFE)

Subfunction	Function Name
254 (0xFE)	Unknown

Page Format Group 208 (0xD0)

Subfunction	Function Name
0	Set Line Height
1	Left/Right Margin Set
2	Spacing Set
3	Hyphenation Zone Set
4	Tab Set
5	Top/Bottom Margin Set
6	Justification *added for WP5.1*
7	Suppress Page Characteristics
8	Page Number Position
11 (0xB)	Form

Font Group 209 (0xD1)

Subfunction	Function Name
0	Color
1	Font Change
2	Color (DrawPerfect) *added for WP5.1*
3	Font Pattern Attributes (DrawPerfect)

Definition Group 210 (0xD2)

Subfunction	Function Name
0	Define Math Columns
1	Define Columns (Newpaper)
2	Paragraph Number Definition
3	Footnote Options
4	Endnote Options
5	Graphics Box Options for Figures
6	Graphics Box Options for Tables
7	Graphics Box Options for Text Boxes
8	Graphics Box Options for User-Defined Boxes
9	Graphics Box Options for Equations *added for WP5.1*
10 (0xA)	Reserved
11 (0xB)	Define Tables (Table Def) *added for WP5.1*
12 (0xC)	Reserved

Subfunction	Function Name
13 (0xD)	Define Link Start *added for WP5.1*
14 (0xE)	Define Link End addid for WP5.1
15 (0xF)	Define DDE Link Start *added for WP5.1*
16 (0x10)	Define DDE Link End *added for WP5.1*
17 (0x11)	Border Options

Set Group 211 (0xD3)

Subfunction	Function Name
0	Set Alignment Character
1	Set Underline Mode
2	Set Footnote Number
3	Set Endnote Number
4	Set Page Number
5	Line Numbering
6	Advance to Page Position
7	Force Odd/Even Page
8	Character at Baseline in Fixed Line Height *added for 6/89 version of WP5.0 and for WP5.1*
9	Reserved
10 (0xA)	Character/Space Width
11 (0xB)	Space Expansion
12 (0xC)	Set Graph Box Number for Figures
13 (0xD)	Set Graph Box Number for Tables
14 (0xE)	Set Graph Box Number for Text Boxes
15 (0xF)	Set Graph Box Number for User-Defined Boxes
16 (0x10)	Set Graph Box Number for Equations *added for WP5.1*
17 (0x11)	Set Language
18 (0x12)	Set Page Number Style *added for WP5.1*
19 (0x13)	Set Direction *Hebrew/Arabic WP5.1 only*

Format Group 212 (0xD4)

Subfunction	Function Name
0	End of Page Function (only group 0 required)
1	Beginning of Line Function *added for WP5.1*
2	Graph Box Information Function
3	Marker for Repositioning
4	Function to Contain Non-Editable/Displayable Text *added for WP5.1*
5	Justification Information *added for WP5.1*
6	Valid On Marker for Undo *added for WP5.1*
7	Valid On Marker for Undo *added for WP5.1*
8	Invalid Function for Undo
9	Display Information for Table, Beginning for Row Codes
14 (0xE)	Data Marker for Intellitag only

Header/Footer Group 213 (0xD5)

Subfunction	Function Name
0	Header A
1	Header B
2	Footer A
3	Footer B

Footnote/Endnote Group 214 (0xD6)

Subfunction	Function Name
0	Footnote
1	Endnote

Generate Group 215 (0xD7)

Subfunction	Function Name
0	Begin Marked Text
1	End Marked Text
2	Define Marked Text
3	Index Entry
4	Table of Authority Entry
5	Endnotes Print Here
6	Save Page Information
7	Auto Reference Definition
8	Auto Reference Tag
9	Include Subdocument
10 (0xA)	Start of Included Subdocument
11 (0xB)	End of Included Subdocument

Display Group 216 (0xD8)

Subfunction	Function Name
0	Date Function
1	Paragraph Number
2	Overstrike
3	Page Number Style Insert *added for WP5.1*

Miscellaneous Group 217 (0xD9)

Subfunction	Function Name
0	Embedded Printer Command
1	Conditional End of Page Function
2	Comment
3	Kerning
4	Outline On *added for WP5.1*
5	Leading Adjustment *added for WP5.1*
6	Kerning
7	Kerning

Subfunction	Function Name
8	Macro *added for WP5.1*

Box Group 218 (0xDA)

Subfunction	Function Name
0	Figure
1	Table
2	Text Box
3	User-Defined Box
4	Equation *added for WP5.1*
5	Horizontal Line
6	Vertical Line

Style Group 219 (0xDB)

Subfunction	Function Name
0	Begin Style ON (paired)
1	End Style ON (paired)
2	Global ON (open style)
3	Style OFF

Table End of Line Codes Group 220 (0xDC)

Subfunction	Function Name
0	Beginning of Column (Cell) at End of Line
1	Beginning of Row at End of Line
2	Table Off at End of Line
3	Reserved

Table End of Page Codes Group 221 (0xDD)

Subfunction	Function Name
0	Reserved
1	Beginning of Row at End of Page
2	Table Off at End of Page
3	Beginning of Row/Hard Page Break

Enhanced Merge Command Codes Group 222 (0xDE)

Subfunction	Function Name
32-99 (0x20-0x63)	Enhanced Merge Command Codes

Equation Nested Function Group 223 (0xDF)

Subfunction	Function Name
0	Equation Nested Function *added for WP5.1*

Tag Group 225 (0xDH) ***for Intellitag only***

Subfunction	Function Name
0	Start Tag
1	End Tag
2	Conversion Rule
3	Ignore On
4	Ignore Off
5	Entity Reference On
6	Entity Reference Off

Unknown Function Group

Unknown is reserved for [UNKNOWN]
Function Code: 254 (0xFE)
Subfunction: 254 (0xFE)
Function Name: Unknown

Structure:	Offset	Size	Meaning
	0	1 byte	Begin function code (254 [0xFE])
	1	1 byte	Begin subfunction code (254 [0xFE])
	2	short	Length (variable)
	4	variable	Data
	..	short	Length (variable)
	..	1 byte	End subfunction code (254 [0xFE])
	..	1 byte	End function code (254 [0xFE])

Font Group (209 [0xD1])

Function Code: 209 (0xD1)
Subfunction: 0
Function Name: Color

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function code (209 [0xD1])
	1	1 byte	Begin subfunction code (0)
	2	short	Length (10 [0xA])
	4	3 bytes	Old print colors (RGB values, 0–0xFF [0 –255])
	7	3 bytes	New print colors (same as above)
	10	short	Length (10 [0xA])
	12	1 byte	End subfunction code (0)
	13	1 byte	End function code (209 [0xD1])

Function Code: 209 (0xD1)
Subfunction: 1
Function Name: Font Change

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function code (209 [0xD1])
	1	1 byte	Begin subfunction code (1)
	2	short	Length (35 [0x23])
	4	1 byte	Old font #
	5	24 bytes	Desired font description (see Font Descriptor, Packet Type 15 [0xF])
	29	1 byte	Matched font # (0 = default)

Note: *Multiply this value by 86 to get the offset into packet type 15 (0xF) to link this code to the font information in the Prefix area. See the information under* WordPerfect Font Procedure *(formatted documents only).*

	30	short	Matched font hash value (0 = default)

Note: *Matches to "Hash of Descriptor" field (see Packet Type 15 [0xF], Offset 21). If the values match, the font information is correct. If the values don't match, the document was saved with the "Fast Save" option set to "Yes," and the correct font cannot be found.*

	32	short	Point size added for WP5.1 (3600ths of an inch)
	34	1 byte	Typeface flags added for WP5.1 (see "Font Descriptor" in Packet Type 15 [0xF])
	35	short	Length (35 [0x23])
	37	1 byte	End subfunction code (1)
	38	1 byte	End function code (209 [0xD1])

Note: *In WP5.0 and in some WP5.1 versions it is necessary to change the Cell Height and Optimal*

Width fields of the Font Descriptor to change a character's point size. To keep full compatibility with WP5.0 and all versions of WP5.1, the Cell Height and Optimal Width must be changed when the point size is changed.

Function Code: 209 (0xD1)
Subfunction: 2
Function Name: Color (DrawPerfect) *added for WP5.1*

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function code (209 [0xD1])
	1	1 byte	Begin subfunction code (2)
	2	short	Length (6)
	4	1 byte	Old print colors (color value, 0–0xFF [0 –255])
	5	1 byte	New print colors (color value, 0–0xFF [0 –255])
	6	short	Length (6)
	8	1 byte	End subfunction code (2)
	9	1 byte	End function code (209 [0xD1])

Function Code: 209 (0xD1)
Subfunction: 3
Function Name: Font Pattern Attributes (DrawPerfect)

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function code (209 [0xD1])
	1	1 byte	Begin subfunction code (3)
	2	short	Length (10 [0xA])
	4	1 byte	Old pattern type (see WPG record type (1) Fill Attribute)
	5	1 byte	Old pattern background color
	6	1 byte	Old pattern foreground color
	7	1 byte	New pattern foreground color
	8	1 byte	New pattern background color
	9	1 byte	New pattern type
	10	short	Length (10 [0xA])
	12	1 byte	End subfunction code (3)
	13	1 byte	End function code (209 [0xD1])

Definition Group (210 [0xD2])

Function Code: 210 (0xD2)
Subfunction: 0
Function Name: Define Math Columns

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function code (210 [0xD2])
	1	1 byte	Begin subfunction code (0)
	2	short	Length (212 [0xD4])
	4	24 bytes	Old math definition (no meaning in new values)
	28	20 bytes	Old calc 0
	48	20 bytes	Old calc 1
	68	20 bytes	Old calc 2
	88	20 bytes	Old calc 3
	108	24 bytes	New math definition Bits 0,1,2 are # of digits, but must be in range of (0–4) bit 3 = 0 use parens for negative #s = 1 use "þ" for negative #s If bit 7 = 0; bits 4-6 = 1 text = 2 numeric = 3 total If bit 7 = 1; bits 4-6 = 0 calc, Formula 1 = 1 calc, Formula 2 = 2 calc, Formula 3 = 3 calc, Formula 4
	132	20 bytes	New calc 0
	152	20 bytes	New calc 1
	172	20 bytes	New calc 2
	192	20 bytes	New calc 3
	212	short	Length (212 [0xD4])
	214	1 byte	End subfunction code (0)
	215	1 byte	End function code (210 [0xD2])

Math Columns
To find the value of a designated column (A–X), take the uppercase ASCII value of the column plus 64 (0x40). For instance, column D is 132 (0x84) (ASCII value 68 [0x44] plus 64 [0x40]).

Formats for the "New Calc" Values

Math Operators

The data are stored in post fix or Reverse Polish Notation (RPN) order. All post fix math rules apply. The operators are stored in the following WordPerfect-specific codes:

Add	(+)	1
Subtract	(þ)	2
Multiply	(*)	3
Divide	(/)	4
Negative	(þ)	5
Positive	(+)	6
Average	(+/)	7
Equal	(=)	8
Average Totals	(=/)	9

Constants

Each (decimal) digit of a constant is stored as a hexadecimal value in a nibble (4 bits, 2 nibbles/byte). The first nibble of the constant equals 12 (0xC) and marks the beginning of the number. Following this nibble, each digit is stored as that digit plus 1 (that is, 0 = 1, 1 = 2, 9 = 10 [0xA]) and occupies one nibble. If the number occupies fewer than seven digits, a 15 (0xF) follows the last digit. A seven-digit number has no 15 (0xF) value on the end. The 14 (0xE) value represents a decimal point. WordPerfect puts 1 into unused bytes that follow 15 (0xF), but these unused bytes are not required in order to read the value of the constant.

Examples:	**Hex Bytes**	**Decimal Value**
	C2 3E 4F O1	12.3
	C1 F0 01 01	0
	C3 51 2F 00	2,401 (in the last unused byte [nibbles], 00 is equivalent to 01)
	CA 11 11 11	9,000,000
	CE 42 71 F0	.3160
	C1 E4 27 1F	0.3160
	CA 9E 87 F0	98.76

Function Code: 210 (0xD2)
Subfunction: 1
Function Name: Define Columns (Newspaper/Parallel)

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function code (210 [0xD2])
	1	1 byte	Begin subfunction code (1)
	2	short	Length (198 [0xC6])
	4	1 byte	Old # of columns (no meaning in new values)
	5	2 *24 short int. Old column left/right margins (third parties fill with nulls)	

Offset	Size	Meaning
101	1 byte	New # of columns bits 0–4 = # of columns (2–24) bit 5 = unused bit 6 = 1 parallel columns bit 7 = 1 parallel columns with block protect
102	2*24 short int.	New column left/right margins (“wpu,” absolute)
198	short	Length (198 [0xC6])
200	1 byte	End subfunction code (1)
201	1 byte	End function code (210 [0xD2])

Function Code: 210 (0xD2)
Subfunction: 2
Function Name: Paragraph Number Definition
Structure:

Offset	Size	Meaning
0	1 byte	Begin function code (210 [0xD2])
1	1 byte	Begin subfunction code (2)
2	short	Length = 140 (0x8C)(84 [0x54] for WP5.0)
4	8*3 bytes	Old 8 definitions (in 3-byte groups)
28	8 short int.	Old 8 level #s
44	8*3 bytes	New 8 definitions Second byte: I = caps Roman I = lowercase Roman A = caps letter a = lowercase letter 1 = Arabic First and third bytes: ASCII punctuation characters. Spaces do not appear in paragraph #s. If the first byte is 0, the second and third bytes are characters to be used as bullets.
68	8 short Int.	New 8-level #s

Added for WP5.1:

84	1 byte	Old “attach” flag If bit set, it means attach previous level
85	1 byte	Old outline flag bit 0 = clear—Enter key inserts paragraph # bit 1 = clear—tab to level of previous paragraph #
86	18 bytes	Old outline style name
104	1 byte	New “attach” flags
105	1 byte	New outline flag

Offset	Size	Meaning
106	18 bytes	New outline style name (if null, not using styles)
124	16 bytes	Old 8-level #s from previous definition
140	short	Length = 140 (0x8C)(84 [0x54] for WP5.0)
142	1 byte	End subfunction code (2)
143	1 byte	End function code (210 [0xD2])

Function Code: 210 (0xD2)
Subfunction: 3
Function Name: Footnote Options
Structure:

Offset	Size	Meaning
0	1 byte	Begin function code (210 [0xD2])
1	1 byte	Begin subfunction code (3)
2	short	Length (160 [0xA0])
4	78 bytes	Old values; new values start at offset 82
82	short	Sets line spacing within the note
84	short	Sets the amount of space between notes (wpu)
86	short	Defines the amount of a footnote to keep together if the footnote is too large to fit on a single page (wpu)
88	1 byte	**Flags:** bits 0–1 = 0 use #s = 1 use characters = 2 use letters bit 2 = 1 numbering starts on each page bit 3 = 1 footnote continued message bit 4 = 0 footnotes after text on page (this option places the footnote after the last line of text on the page) = 1 footnotes at bottom of page (this option places footnotes at the bottom of the page without regard to the amount of text on the page) bits 5,6 = 0 no line drawn between text and footnotes = 1 2þ line drawn between text and footnotes = 2 lines drawn from left to right margin, between text and footnotes
89	1 byte	# of chars used in place of footnote #s
90	5 short int.	Characters used in place of footnote #s (null terminated if < 5)
100	20 bytes	String for footnote # in text (attribute)
120	20 bytes	String for footnote # in note (attribute)
140	short	Left margin width for footnotes (wpu)
142	short	Right margin width for footnotes (wpu)

Offset	Size	Meaning
144	short	Line height for footnotes (wpu) (refer to function 208 [0xD0] subfunction 0)
146	short	Character width for footnotes (wpu) (211 [0xD3] sub 10 [0xA])
148	short	Space width for footnotes (refer to 211 [0xD3] sub 10 [0xA])
150	short	% space minimum for footnotes
152	short	% space maximum for footnotes

Note: *If you are creating this function code, offsets 144–152 can be left blank for WordPerfect to fill in later.*

154	short	Attribute for footnotes (195 [0xC3])
156	3 bytes	Color for footnotes (209 [0xD1])

Note: *If you are creating this function code, offset 156 can be left blank for WordPerfect to fill in later.*

159	1 byte	Font for footnotes

Note: *This value works exactly the same as the "match font number" in the font change code. See* WordPerfect Font Procedure *for more information.*

160	short	Length (160 [0xA0])
162	1 byte	End subfunction code (3)
163	1 byte	End function code (210 [0xD2])

Function Code: 210 (0xD2)
Subfunction: 4
Function Name: Endnote Options

Structure:

Offset	Size	Meaning
0	1 byte	Begin function code (210 [0xD2])
1	1 byte	Begin subfunction code (4)
2	short	Length (160 [0xA0])
4	78 bytes	Old values

Note: *New values begin for 78 bytes.*

82	short	Spacing in endnotes
84	short	Spacing between endnotes
86	short	Amount of endnote to keep together

Offset	Size	Meaning
88	1 byte	**Flags:** (bits: xx xx xx en) (xx = not used) en: = 0 use #s = 1 use characters = 2 use letters
89	1 byte	# of chars used in place of endnote #s
90	5 short int.	Characters used in place of endnote #s (null terminated if < 5)
100	20 bytes	String for endnote # in text
120	20 bytes	String for endnote # in note
140	short	Left margin for endnotes (wpu)
142	short	Right margin width for endnotes (wpu)
144	short	Lines/inch for endnotes
146	short	Character width for endnotes; 0 = auto
148	short	Space width for endnotes
150	short	% space minimum for endnotes
152	short	% space maximum for endnotes
154	short	Attribute for endnotes
156	3 bytes	Color for endnotes
159	1 byte	Font for endnotes
160	short	Length (160 [0xA0])
162	1 byte	End subfunction code (4)
163	1 byte	End function code (210 [0xD2])

Function Code: 210 (0xD2)
Subfunction: 5
Function Name: Graphics Box Options for Figures
Structure:

Offset	Size	Meaning
0	1 byte	Begin function code (210 [0xD2])
1	1 byte	Begin subfunction code (5)
2	short	Length (128 [0x80])
4	62 bytes	Old values (no meaning in new values)
66	62 bytes	New values
	1 byte	**Flags:** bits 0–1 numbering style for box #, level 1 = 1 use #s = 2 use letters (uppercase) = 3 use Roman numerals (uppercase) bits 2,3 numbering style for box #, level 2 = 0 this level not used = 1 use #s = 2 use letters (lowercase) = 3 use Roman numerals (lowercase)

Offset	Size	Meaning
		bit 4
		= 0 position caption below window (or left-centered for equations if inside) *added for WP5.1*
		= 1 position caption above window (or rightcentered for equations if inside) *added for WP5.1*
		bit 5
		= 0 position caption outside borders (above or below for equations) *added for WP5.1*
		= 1 position caption inside windows (for equations, signals left or right) *added for WP5.1*
	1 byte	Shading % (0 –100%, 0 = no shade)
	short	Border styles (4 bits each: Left Right Top Bottom) (high to low)
		= 0 none
		= 1 single
		= 2 double
		= 3 dashed
		= 4 dotted
		= 5 thick
		= 6 extra thick
		= 7 not used
	short	Minimum offset from start of paragraph (wpu)
	4 short int.	Spacing between border and text (wpu), 1 short integer each: left, right, top, and bottom
	4 short int.	Spacing between border and figure (wpu), 1 short integer each: left, right, top, and bottom
	20 bytes	String for box # in caption
	short	Old box # (no meaning in new values)
	short	Line spacing for captions (binary point between bytes)
	short	Line height for captions
	short	Character width for captions, 0 = auto (wpu)
	short	Space width for captions (wpu)
	short	% space minimum for captions
	short	% space maximum for captions
	short	Attribute for captions (195 [0xC3])
	3 bytes	Color for captions (209 [0xD1])
	1 byte	Font for captions
128	short	Length (128 [0x80])
130	1 byte	End subfunction code (5)
131	1 byte	End function code (210 [0xD2])

Function Code: 210 (0xD2)
Subfunction: 6

Function Name: Graphics Box Options for Tables
Structure: See *Graphics Box Options for Figures*

Function Code: 210 (0xD2)
Subfunction: 7
Function Name: Graphics Box Options for Text Boxes
Structure: See *Graphics Box Options for Figures*

Function Code: 210 (0xD2)
Subfunction: 8
Function Name: Graphics Box Options for User-Defined Boxes
Structure: See *Graphics Box Options for Figures*

Function Code: 210 (0xD2)
Subfunction: 9
Function Name: Graphics Box Options for Equations *added for WP5.1*
Structure: See *Graphics Box Options for Figures*

Function Code: 210 (0xD2)
Subfunction: 10 (0xA) Reserved

Function Code: 210 (0xD2)
Subfunction: 11 (0xB)
Function Name: Define Tables (Table Def) *added for WP5.1*
Structure:

Offset	**Size**	**Meaning**
0	1 byte	Begin function code (210 [0xD2])
1	1 byte	Begin subfunction code (11 [0xB])
2	short	Length (variable)
4	1 byte	**Old Flags:** table bits 0–2 table position options: = 0 align with left margin = 1 align with right margin = 2 center between margins = 3 full (adjust column widths to fit margins) = 4 absolute offset from left margin (set position) bits 3,4 available bit 5 = 0 minus signs = 1 display negative results with

parentheses

Offset	Size	Meaning
		bit 6 = 1 auto adjust column widths to fit margins bit 7 = 1 expand column widths to fit margins if bit 6 is set
5	1 byte	Old shading % (0 –100)
6	short	Old # of columns (max = 32 [0x20])
8	short	Old table # (same # sequence as table boxes, set by formatter)
10	short	Old position of left edge of table (wpu) (relative to left margin)
12	short	Old left gutter space (wpu)
14	short	Old right gutter space (wpu)
16	short	Old top gutter space (wpu)
18	short	Old bottom gutter space (wpu)
20	short	Old row # after header rows (0 = no header rows)
22	short	Old formatter lines at start of table (wpu) (used to calculate size of header rows)
24	short	Old page # at start of table (used to calculate size of header rows in formatter)
26	short	Old offset from left edge of paper (for absolute position option)
28	1 short int./ column	Old column widths Old widths in wpu, left of left border to left of right border; width includes left border width and gutter space—widths in wpu
..	1 short int./ column	Old column attributes 1 = extra large 2 = very large 4 = large 8 = small 16 (0x10) = fine 32 (0x20) = superscript 64 (0x40) = subscript 128 (0x80) = outline 256 (0x100) = italics 512 (0x200) = shadow Offset 1024 (0x400) = redline 2048 (0x800) = double underline 4096 (0x1000) = bold 8192 (0x2000) = strike-out 16384 (0x4000) = underline 32768 (0x8000) = small caps

Offset	Size	Meaning
..	1 byte/	Old column alignmentscolumn Old bits 0–2 horizontal alignment for column = 0 left = 1 flush right *modified for WPWin 5.1* = 2 center = 3 justified (left & right) *modified for WPWin 5.1* = 4 decimal align = 5 (available) = 6 (available) = 7 (available) bit 3 available bits 4–7# of chars to right of decimal alignment for align cells
..	variable	New values that repeat old values from "flags" at offset 4 through "1 byte/column, column alignments." To calculate the offset of new values, use the following formula: 4 bytes (first 4 bytes of function) + 24 bytes (fixed # of old values) + 5 (# of columns) (old values that vary according to # * of columns of default or previous table) = start of new values
..	2 bytes	Reserved
..	short	Length (variable)
..	1 byte	End subfunction code (11 [0xB])
..	1 byte	End function code (210 [0xD2])

Function Code: 210 (0xD2)
Subfunction: 12 (0xC) Reserved

Function Code: 210 (0xD2)
Subfunction: 13 (0xD)

Function Name: Define Link Start added for WP5.1

Structure:

Offset	Size	Meaning
0	1 byte	Begin function code (210 [0xD2])
1	1 byte	Begin subfunction code (13 [0xD])
2	short	Length (variable)
4	short	Old screen column (su)
6	1 byte	# of lines of text to display

	Offset	Size	Meaning
	7	1 byte	Type (spreadsheet text or table, database, graphic, etc.)
	8	short	Date of file when last linked
	10	short	Time of file when last linked
	12	short	Beginning column of range
	14	short	Beginning row of range
	16	short	Ending column of range
	18	short	Ending row of range
	20	81 bytes max	Filename with len byte (max 81)
	..	21 bytes max	Range name with len byte (max 21)
	..	16 bytes max	Range reference with len byte (max 16)
	..	short	Length (variable)
	..	1 byte	End subfunction code (13 [0xD])
	..	1 byte	End function code (210 [0xD2])

Function Code: 210 (0xD2)
Subfunction: 14 (0xE)
Function Name: Define Link End *added for WP5.1*

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function code (210 [0xD2])
	1	1 byte	Begin subfunction code (14 [0xE])
	2	short	Length (variable)
	4	short	Old screen column (su)
	6	1 byte	# of lines of text to display
	7	short	Beginning column of range
	9	short	Beginning row of range
	11	short	Ending column of range
	13	short	Ending row of range
	14	81 bytes max	Filename with len byte (max 81)
..	short	Length (variable)	
	..	1 byte	End subfunction code (14 [0xE])
	..	1 byte	End function code (210 [0xD2])

Function Code: 210 (0xD2)
Subfunction: 15 (0xF)
Function Name: Define DDE Link Start *added for WPWin 5.1*

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function (210 [0xD2])
	1	1 byte	Begin Subfunction (15 [0xF])
	2	short	Length (variable)
	4	short	Run-time link ID. Unique ID valid while document is open.

Offset	Size	Meaning

Note: *Developers should enter null to initialize.*

6	1 byte	Link Options bits 0-1: Storage preference 1 = 0 invalid = 1 text = 2 graphics = 3 reserved bits 2-3: Storage preference 2 = 0 no second choice = 1 text = 2 graphics = 3 reserved bits 4-5: Update mode = 0 invalid = 1 manual = 2 automatic = 3 reserved bits 6-7: Reserved, must be 0
7	4 bytes	Data Time Stamp (seconds) from the beginning of year, according to the following assumptions: Sec (0–59) (60 per min) Min (0–59) (60 per hr) Hr (0–23) (24 per day) Day (1–31) (31 per mon) Mon (1–12) The time is 0 if WPWin hasn't received data from the source file.
11	short	Data Time Stamp (year) (that is, 1992); the time is 0 if WPWin hasn't received data from the source file.
13	variable	Link name. Null terminated WP string chosen by user. (A WP string may have null bytes within extended character codes.) The name identifies the link to the user in WPWin dialog boxes.
..	variable	Link Source. Null-terminated ANSI string giving source application, file (topic), and item names separated by vertical bars (|). The second vertical bar and the item name may be omitted if the link refers to the entire source file.
..	short	Length (variable)
..	1 byte	End subfunction (15 [0xF])
..	1 byte	End function (210 [0xD2])

Function Code: 210 (0xD2)
Subfunction: 16 (0x10)
Function Name: Define DDE Link End *added for WPWin 5.1*

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function (210 [0xD2])
	1	1 byte	Begin subfunction (16 [0x10])
	2	short	Length (6)
	4	short	Run-Time Link ID. This unique ID is valid while the document is open. The ID is 0 if WPWin hasn't saved the document with this link in it.
	6	short	Length (6)
	8	1 byte	End subfunction (16 [0x10])
	9	1 byte	End function (210 [0xD2])

Function Code: 210 (0xD2)
Subfunction: 17 (0x11)
Function Name: Border Options

Structure:	**Offset**	**Size**	**Meaning**
	0	1 byte	Begin function (210 [0xD2])
	1	1 byte	Begin subfunction (17 [0x11])
	2	short	Length (57 [0x39])
	4	1 byte	Size of other information (0)
	5	26 bytes	Old border information (6 * 3) + 8 bytes

Note: *Beginning of new border information.*

31	short	Single width
33	1 byte	Single shade
34	short	Double width
36	1 byte	Double shade
37	short	Dashed width
39	1 byte	Dashed shade
40	short	Dotted width
42	1 byte	Dotted shade
43	short	Thick width
45	1 byte	Thick shade
46	short	Extra thick width
48	1 byte	Extra thick shade
49	short	Double inside spacing
51	short	Dash spacing
53	short	Dash length
55	short	Dot spacing
57	short	Length (57 [0x39])

Offset	Size	Meaning
59	1 byte	End subfunction (17 [0x11])
60	1 byte	End function (210 [0xD2])