MICROSOFT OFFICE WORD 97-2007 BINARY FILE FORMAT SPECIFICATION [*.doc]

Includes Binary File Format Documentation

Relevant To:

Microsoft Office Word 2007

Microsoft Office Word 2003

Microsoft Office Word 2002

Microsoft Office Word 2000

Microsoft Office Word 1997

Microsoft Office Word 97-2007 Binary File Format (.doc) Specification

NOTICE

Table of Contents

Note

Many of the structures written into Microsoft® Office Word 2007 .doc files differ slightly from the corresponding structures Word uses internally. The file-specific version of a structure is typically named by adding a preceding or (more often) trailing "F". For example, Word uses a `PLC` (**PL**ex of **C**ps (Character positions)) internally, but writes to files a `PLCF` (**PL**ex of **C**ps in **F**ile). Many discussions in this document use the name of the internal structure when the file-specific structure is what is really being referred to. The reader should remember that the name of a seemingly undefined structure type may simply be missing a leading or trailing "F".

Additions to Word 2007

There were several additions to the binary file format with the release of Microsoft Office Word 2007. Word 2007 introduces a new XML-based file format. While this new format is the default format for documents saved by Word 2007, Word 2007 also provides the capability to save files to the binary Word file format used in previous versions.

Additions to the Word binary file format include the following:

- Addition of a new custom XML storage
- Addition of several new records to the Office Drawing Layer (Escher)

Word 2007 adds a new custom XML storage to the binary file format to store custom XML information for documents created in Word 2007. This release of the Word binary file format documentation includes information about the custom XML storage. The additions to this binary file format documentation are preceded with the note: "Added in Word 2007".

Word 2007 adds several new records to the binary file format to store information about documents created in Word 2007. Each of these records stores information about features specific to Word 2007. This data is preserved in the binary format so that when reopened in Word 2007, documents retain data and features that are only available in the newer version. Information on these new records to the Office Drawing Layer is documented in the Office Drawing Binary File Format documentation.

Word and .Doc Files

The binary format for Microsoft Word 97 and later versions is based on a structure referred to as a .doc file or "compound file". Compound File Binary format is documented at

http://download.microsoft.com/download/0/B/E/0BE8BDD7-E5E8-422A-ABFD-4342ED7AD886/WindowsCompoundBinaryFileFormatSpecification.pdf.

A Word .doc file consists of a:

- Main stream
- Summary information stream
- Table stream
- Data stream
- Custom XML storage *(Added in Word 2007)*
- 0 or more object streams which contain private data for OLE 2.0 objects embedded within the Word document

The summary information stream is described in the **Format of the Summary Info Stream in a Word File** section.

The object stream contains binary data for embedded objects. Word has no knowledge of the contents ofthis stream.

The majority of this document describes the contents of the main stream and the table stream.

Definitions

API (Application Programming Interface):
A set of libraries, functions, definitions, etc. providing an interface to a programming environment or model.

bin table
Each FKP can be viewed as a bucket or **bin** that contains the properties of a certain range of FCs in the Word file. In Word files, a PLC, the plcfbte (**PL**ex of F**C**s containing **B**in **T**able **E**ntries) is maintained. It records the association between a particular range of FCs and the PN (**P**age **N**umber) of the FKP that contains the properties for that FC range in the file. In a **complex** (fast-saved) Word document, FKP pages are intermingled with pages of text in a random pattern which reflects the history of past fast saves. In a complex document, a plcfbteChpx which records the location of every CHPX FKP must be stored and a plcfbtePapx which records the location of every PAPX FKP must be stored. In a **non-complex**, full-saved document, all of the CHPX FKPS are recorded in consecutive 512-byte pages with the FKPs recorded in ascending FC order, as are all of the PAPX FKPS. A plcfbteLvcx serves the same purpose for LVCX FKPS.

In a full save document, the plcfbte's may not have been able to expand during the save process due to a lack of RAM. In this situation, the plcfbte's are interspersed with the property pages in a linked list of FBD pages.

bookmark
A **bookmark** associates a user defined name to a range of text within a document. A bookmark is frequently used as an operand in **field code** instructions within a field. A bookmark is represented by three parallel data structures, the sttbBkmk, the plcbkf and the plcbkl. The sttbBkmk is a string table which contains the name of each defined bookmark. The plcbkf records the beginning CP position of each bookmark. The plcbkl records the limit CP position that delimits the end of a bookmark. Since bookmarks may be nested within one another to any level, the BKF structure stored in the plcbkf consists of a single index that identifies which plcbkl marks the end of the bookmark. The BKL structure is not written to the file and the plcbkl contains only CPs.

character style
A named character property exception that can be associated with any number of runs of text in a Word document's text stream. When a run of text is tagged with a particular **character style**, a chpx defined for the character style is applied to the character properties defined for the paragraph style of the paragraph that contains the text. This means that the character style can change one or more of the character property field settings specified by the paragraph style of a paragraph to a particular setting without changing the value of any other field.

CHP (CHaracter Properties)
The data structure describing the character properties of a run of text.

CHPX (Character Property EXception)
A data structure describing how a particular CHP differs from a reference CHP. In Word 6.0, the CHPX simply consists of a grpprl applied to the reference CHP to produce the originally encoded CHP. By applying a CHPX to the character properties (CHP) inherited by a particular paragraph from its **style**, it is possible to reconstitute the CHP for the portion of the character run that intersects that paragraph.

COLORREF
Used to specify an explicit RGB color, the COLORREF value has the following hexadecimal form:

0x00bbggrr

The low-order byte contains a value for the relative intensity of red, the second byte contains a value for green, and the third byte contains a value for blue. The high-order byte must be zero. The maximum value for a single byte is 0xFF.

The intensity for each argument is in the range 0 through 255. If all three intensities are zero, the result is black. If all three intensities are 255, the result is white.

CP (Character Position):
A four-byte integer specifying the position coordinate of a character of text within the logical text stream of a document.

Custom XML Datastore (*Added in Word 2007*):
The custom XML data store specifies custom defined XML files contained in the binary Microsoft Word 97 format or the Office Open XML Formats.

data stream:
The stream within a Word .doc file containing various data that anchors to characters in the main stream. For example, binary data describing in-line pictures and/or form fields.

docfile:
An OLE 2.0 compatible multi-stream file. Word files are .doc files.

document:
A named, multi-linked list of data structures, representing an ordered stream of text with properties produced by a user of Microsoft Word.

DOP (DOcument Properties)
The data structure describing properties that apply to the document as a whole.

Dxas

embedded object
The native data for embedded objects (OBJs) is stored similarly to pictures (`PIC`s). To locate the native data for Embedded objects, scan the `plc` of field codes for the mother, header, footnote and annotation, textbox and header textbox documents (`fib.PlcffldMom/Hdr/Ftn/Atn/Txbx/HdrTxbx`). For each separator field, get the `chp`.

If `chp.fSpec=1` and `chp.fObj=1`, then this separator field has an associated embedded object. The file location of the object data is stored in `chp.fcObj`. At the specified location an object header is stored followed by the native data for the object. See the `_OBJHEADER` structure.

If `chp.fOle2=1`, then this separator field has an associated OLE2 object. The `fcPic` will be a unique integer that specifies the name of the object's sub-storage instead of an offset into the data stream.

fast-saved (or complex) file:
A Word file in which the physical order of characters stored in the file does not match the logical order of characters in the document that the file represents. A **piece table** must be stored in the file to describe the text stream of the document. Due to Unicode compression to code page 1252, all files (simple and complex) now contain a piece table.

FC(File Character position):
A four-byte integer which is the byte offset of a character (or other object) from the beginning of a stream of the .doc file. Before a file has been edited (i.e. in a full saved Word document), `CP`s can be transformed into `FC`s by adding the `FC` coordinate of the beginning

of a document's text stream to the CP. After a file has been edited (i.e. in a fast-saved Word document), the mapping from CP to FC is recorded in the **piece table** (see below).

FIB (File Information Block):

The header of a Word file. Begins at offset 0 in the file. This gives the beginning offset and lengths of the document's text stream and subsidiary data structures within the file. Also stores other file status information.

field

A field is a two-part structure that may be recorded in the CP stream of a document. The first part of the structure contains **field codes** which instruct Word to insert text into the second part of the structure, the **field result**. Fields in Word are used to insert text from an external file or to quote another part of a document, to mark index and table of contents entries and produce indexes and tables of contents, maintain DDE links to other programs, to produce dates, times, page numbers, sequence numbers, etc. There are 91 different field types.

A **field begin mark** delimits the beginning of a field and precedes any of the field codes stored in the field. The end of the field codes and the beginning of the field result is marked with the **field separator** and the field result and the field itself are terminated by a **field end mark**.

The CP locations of the field begin mark, field separator, and field end mark are recorded in plcfld data structures that are maintained for the main document and all of the subdocuments of the main document whenever a field is inserted or edited. A field can be **dead**, in which case it has no field separator, no field result, and no entry in the plcfld. (See the definition of the FLD structure for a list of possible dead field code strings.) An array of two-byte FLD structures is stored in the plcfld in a 1-to-1 correspondence with the recorded CP entries. An FLD associated with a **field begin mark** records the type of the field. An FLD associated with the **field end mark** records the current status of the field (i.e. whether the result is dirty or has been edited, whether the result has been locked, etc.)

Fields may be nested. Twenty (20) levels of nesting are permitted.

FKP (Formatted disK Page):

A data structure that fits in one 512-byte page that encodes either the character properties or the paragraph properties of a certain portion of a Word .doc file. An FKP consists of four components:

1) a count of the number of runs or paragraphs described by the page.

2) an array of FCs recorded in ascending order demarcating the boundaries between runs or paragraphs that are recorded adjacent to one another in the Word file.

3) In **character** FKPs an array of offsets within the FKP in one to one correspondence with the array of FCs that locate the properties of the run that begins at a particular FC.

In LVC FKPs an array of offsets within the FKP in 1-to-1 correspondence with the array of FCs that locate the LVCXs that describe the run that begins at a particular FC.

In **paragraph** FKPs an array of BX structures follows the array of FCs in 1 to 1 correspondence with the array of FCs. Each BX begins with an offset that locates the properties of the paragraph that begins at a particular FC. The remainder of the BX contains a PHE structure that encodes information about the height of the paragraph that begins at that FC.

4) a group of CHPXs if the FKP stores character properties, a group of PAPXs if the FKP stores paragraph and table properties, or a group of LVCXs if the FKP stores paragraph level and numbering cache information.

To find the CHPX/PAPX corresponding to a particular character in a document, calculate the FC coordinate for that character. Then search through the **bin table** for the type of property you want to produce, to find the FKP in the document stream whose array of FCs encompasses the FC of the document character.

Then search within the FKP to find the index of the largest FC entry that is less than or equal to the FC of the document character. Use this index to look up an offset in the array of offsets (for **character** FKPs) or look up an offset in the array of BXs (for **paragraph** FKPs) within the FKP. Add this offset to the beginning address of the FKP in memory. This will be the first byte of the desired CHPX/PAPX.

full-saved (or non-complex) file:
A Word file in which the physical order of characters stored in the file is identical to the logical order of characters in the document that the file represents. The text stream of a non-complex file can be described by an fc (an offset from the beginning of the file) to mark where the text begins and a ccp (**c**ount of **CP**s) to record how many characters are stored in the text stream. Due to Unicode compression to code page 1252, all files (simple and complex) now contain a piece table. However, a full-saved piece table will not have property modifiers (prms) and all text in the file is referenced by the piece table.

grpprl (group of prls):
A grpprl is a data structure that records a set of sprms. The 0th sprm is recorded at offset 0 of the structure. Any succeeding sprms are recorded immediately after the end of the preceding sprm. To traverse a grpprl and locate the sprms recorded within it, it's necessary to fetch the opcode of the first sprm, lookup the length of the sprm with that opcode, use that length to skip past the first sprm, fetch the opcode of the second sprm, lookup the length of that sprm, use the length to skip the second sprm, and so on. See the table in the "**SPRM Definition**" topic to determine the length of a sprm.

The phrase "**apply the sprms of a grpprl** (or **papx** or **sepx)**" used later in this document means to fetch the 0th sprm recorded in the grpprl and perform the action for that sprm, fetch the first sprm and perform its action, and continue this procedure until all sprms in the grpprl (or papx or sepx) have been processed.

LFO
LFO stands for List Format Override.

LFOLVLF
LFOLVLF stands for List Format Override for a Single Level.

LID
LID stands for Language ID.

LSD
LSD Stands for Latent Style Data.

LST
The LST structure is where most of the list appearance data is stored

LSTF
LST structure on File

LVC
List Level Cache information

LVCX
List Level Cache information

LVL
List Level

LVLF
List Level on File

main stream:
The stream within a Word .doc file containing the bulk of Word's binary data.

object storage:
A storage containing binary data for an embedded OLE 2.0 object. Multiple instances are referred to as "storages".

OLE 2.0:
Object Linking and Embedding 2.0.

Office Drawing object
An Office Drawing object is represented in the document stream as a special character, an ASCII 8, which has `chp.fSpec=1` for the run of text containing the character. Only main documents and header documents contain Office Drawing objects. The native data for an Office Drawing object may be obtained by taking the `CP` for the special character and using this to find the corresponding entry in the `plcspa`. An entry in this `plc` consists of a `FSPA` structure, which is described elsewhere in this document.

Office Drawing objects can have text attached to them. Text for the textboxes is stored separately in the textbox subdocument of the main or header document. The textbox subdocument contains a `plctxbxs` where the text from `CP n` to `CP n+1` in the subdocument is the text which is contained in a textbox as specified in the `TXBXS` structure for this n^{th} entry in the `plctxbxs`. Textboxes can be linked in chains of up to 32 textboxes. Ordering of textboxes in the subdocument is completely unrelated to the document structure due to the nature of textbox linking. To find the text for a given Office Drawing object, the `TXID` property (a long: high word is `itxbxs+1`, low word is the sequence number) must be fetched from the Office art data for the shape. This contains an index (`itxbxs`) into `plctxbxs` and a sequence number in the chain of linked textboxes. The text for the entire chain of linked textboxes is stored from the `CP itxbxs` to `CP itxbxs+1` of `plctxbxs`. The `plctxbxBkd` describes the "page table" within textbox stories (where the textboxes in each linked textbox chain are thought of as "pages"). So, for each entry in the `plctxbxs` there is a corresponding entry in the `plctxbxBkd` at the same `CP`, and there may be additional entries in the `plctxbxBkd` to describe the breaks from one textbox to the next in linked textbox chains.

page (or sector):
A 512 byte segment of the main stream within a Word .doc file that begins on a 512-byte boundary. (bytes 0-511 are in page 0, bytes 512-1023 are in page 1, etc.). In Word data structures, an unsigned two-byte integer page number is given the acronym `PN` (for **P**age **N**umber).

PAP (PAragraph Properties)
The data structure which describes the properties of a particular paragraph.

PAPX (PAragraph Property EXception)
A data structure describing how a particular paragraph's properties differ from the paragraph properties of the style assigned to the paragraph. By applying a `PAPX` to the paragraph properties (`PAP`) inherited by a particular paragraph from its **style**, it is possible to reconstitute the `PAP` for that paragraph. The `PAPX` contains an `ISTD` (a style code to identify the style in control of the paragraph and a `grpprl` which specifies how the style's paragraph properties must be changed to produce the paragraph properties of the paragraph.

paragraph

A contiguous sequence of characters within the text stream of a document that is delimited by a paragraph mark, cell mark, row mark, or a section mark (these are special characters described later in this document).

paragraph style

A named set of character and paragraph properties that can be associated with any number of **paragraphs** in a Word document's text stream. A **paragraph style** provides a set of character and paragraph property defaults for the text of any paragraph tagged with that style. When a new paragraph is created and given a particular style, newly typed text is set to the character and paragraph properties of that style unless the user makes an exception to the paragraph style definition by performing other editing operations.

picture

A picture is represented in the document text stream as a special character, an ASCII 1 whose CHP has the fSpec bit set to 1. The file location of the picture in the Word binary file is stored in the character's CHP in chp.fcPic. The fcPic is a byte offset into the data stream. Beginning at the position recorded in chp.fcPic, a header data structure, the PIC, will be stored. If the picture is a reference to a TIFF file, a Picture file or an Office shape file, the name of the file is recorded immediately following the PIC in a Pascal style string. If the picture is an Office shape, a Window's metafile or a bitmap, the shape, metafile or bitmap will immediately follow the PIC. Pictures that are a reference to an Office shape file will include both the filename and the shape in that order. Pictures inserted with Word 97 and later versions are in the new Office shape format (documented elsewhere). However, pictures can be copied from older files into newer ones and their old format will persist until the picture is edited or displayed**.**

Some files (including all files created by Word for the Macintosh) may store Macintosh PICT pictures as well. In this case, the PIC structure is immediately followed by a standard Windows metafile depicting a large "x", so that older readers expecting only a metafile after the PIC will just display this "x". If a reader detects this standard "x" metafile, it can extract the sizes of the standard "x" metafile and the Macintosh PICT picture that follows it from an early portion of this "x" metafile. See Appendix B for a discussion of this technique.

piece table:

The **piece table** is a data structure that describes the logical sequence of characters in a Word document and records recent changes to the formatting of a Word document. It is stored in a Word file as a PLCF named the plcfpcd (**PL**ex of **C**ps containing **P**ie**c**e **D**escriptors). The piece table relates a logical character number, called a CP (**C**haracter **P**osition), to a physical location within a Word file (an FC). The array of CPs in the plcfpcd defines a partitioning of the Word document into disjoint pieces. The second array is an array of PCDs (**P**ie**c**e **D**escriptors) which is in 1-to-1 correspondence to the array of CPs that records the physical location in the Word file where the corresponding piece begins. To find the physical location of a particular logical character in a Word document, take the CP coordinate of that character within the document and find the piece that contains that character. This is done by finding the index of the largest CP in the array of CPs that is less than the character CP. Then reference the PCD with that index in the array of PCDs. The FC stored in the PCD gives the position of the beginning of the piece in the file. Finally, add the offset of the desired character from the beginning of its piece to the FC of the beginning of the piece. This gives a "virtual" file offset of the character. If the second most significant bit is clear, then this indicates the actual file offset of the Unicode character (two bytes). If the second most significant bit is set, then the actual address of the codepage-1252 compressed version of the Unicode character (one byte), is actually at the offset indicated by clearing this bit and dividing by two.

PL

PLCF(PLex of Cps(or FCs) stored in File):
A data structure consisting of two parallel arrays that allows a relation to be established between a certain CP position in the document text stream (or FC position in a file) and an arbitrary data structure. It consists of an array of n+1 CPs or FCs followed by an array of **n** instances of a particular arbitrary data structure. In typical usage, the **nth** CP or FC of the PLCF is in 1-to-1 correspondence with the **nth** instance of the arbitrary data structure, with the n+1st CP or FC marking the limit of the **nth** instance's influence. When a PLCF is used to record a partitioning of the document's text stream or a partitioning of the bytes stored in a file, the 0th CP/FC stored in the PLCF will be 0. When a PLCF is used to record the location of certain marks or links within the document text stream, the 0th CP/FC stored in the PLCF will record the position of the 0th mark or link. To properly interpret a PLCF stored in a Word file, the length of the stored PLCF and the length of the arbitrary data structure stored in the PLCF must be known. The length of the stored PLCF is recorded in the FIB. The lengths of the data structures stored in PLCFs within Word files are listed later in this document.

PLF(PLex stored in File):
A data structure consisting of an array of structures preceded by a long count of structures.

prm (PRoperty Modifier):
A field in piece table entries that records how the properties of text within a piece were changed to reflect user formatting operations. The prm usually contains an index to a grpprl which records the user's formatting changes as a group of sprms. If the user has made only a small change to formatting that can be expressed as a single 1 or 2-byte sprm, that sprm is stored within the prm.

run of text
A contiguous sequence of characters within the text stream of a document that have the same character formatting properties. A single run may cross paragraph boundaries and may encompass the entire document.

section
A contiguous sequence of paragraphs within the text stream of a document that is delimited by a section mark or by the final paragraph mark at the end of a document. Users frequently treat sections as the equivalent of a chapter in a book. The boundaries of sections mark locations where the layout rules for a document (number of columns, text of headers and footers to use, whether page numbers should be displayed, etc.) are changed.

SEP(SEction Properties)
The data structure describing the properties of a particular section.

SEPX(SEction Property EXceptions)
A data structure describing how the properties of a particular section differ from a Word-defined standard SEP. As in the PAPX, the differences between the SEP for a section and the standard SEP are encoded as a list of sprms that describe how the standard SEP can be transformed into the section's SEP. By applying a SEPX's sprms to the standard SEP, it is possible to reconstitute the SEP for that section.

The PLCFSED, a data structure stored in a .doc file, records the locations of all SEPXs. The array of CPs in the plcfsed records the boundaries of sections in the Word document. The second array in the plcf, an array of SEDs (**SE**ction **D**escriptors), is in 1-to-1 correspondence to the array of CPs. Each SED stores the beginning FC of the SEPX that records the properties for a section. If the FC stored in a SED is -1, the section properties of the section are exactly equal to the standard section properties.

The SEP for a particular section may be constructed if a CP of a character in that section is known. First search the array of CPs in the PLCSED for the index of the largest CP that is less than or equal to the CP of the character. Use this index to locate the SED in the plcfsed which describes the section. The FC stored in the SED is the offset from the beginning of the Word file at which the SEPX is stored. If the stored FC is equal to 0xFFFFFFFF, then the SEP for the section is exactly equal to the standard SEP (see SEP structure definition). Otherwise, read the SEPX into memory and create a copy of the standard SEP. Finally, apply the sprms stored in the SEPX to the standard SEP to produce the SEP for a section.

SPLS

sprm (Single PRoperty Modifier):

An instruction to modify one or more properties within one of the property defining data structures (CHP, PAP, TAP, SEP, or PIC). It consists of an operation code which identifies the field(s) to be changed, and an operand which gives the value that a particular field is changed to or a parameter passed to a procedure to change the field or fields. A prl (**pr**operty modifiers stored in a **l**ist) is a sprm plus its operand.

stream:

The physical encoding of a Word document's text and sub data structures in a random access stream within a .doc file.

STSH (STyle SHeet)

A data structure which represents every style defined within the Word document. The STSH records a unique name string for every style and associates each name with a particular CHP and/or a PAP. The indexes used to refer to individual styles are called ISTDs (**I**ndexes to **ST**yle **D**escriptors). Every PAPX for every paragraph recorded in a document contains an ISTD which identifies the style from which a paragraph inherited its default character and paragraph properties. CHPXs recorded for the text within the paragraph and PAPXs recorded for the paragraph itself encode changes that the user has made with respect to the style's default properties.

STTBF (STring TaBle stored in File)

Word has many tables of strings that are stored as Pascal-type strings. STTBFs consist of an optional short containing 0xFFFF, indicating that the strings are extended character strings, a short indicating how many strings are included in the string table, another short indicating the size in bytes of the extra data stored with each string and each string followed by the extra data. Non-extended character Pascal strings begin with a single byte length count which describes how many characters follow the length byte in the string. If pst is a pointer to an array of characters storing a Pascal style string then the length of the string is *pst+1. In an STTBF Pascal-type strings are concatenated one after another until the length of the STTBF recorded in the FIB is exhausted. Extra data associated with a string may also be stored in an sttbf. When extra data is stored for an STTBF, it is written at the end of each string. For example: the extra data for an STTBF consists of a short. If the string "Cat" were stored, the actual entry in the string table would have a length byte of 3 (3 for "Cat") followed by the bytes 'C' 'a' 't', followed by the 2 bytes containing the short. Extended character strings are stored just the same, except they have a double byte length count and each extended character occupies two bytes.

subdocument

A separate logical stream of text with properties for which correlations with the main document text are maintained. Word's headers/footers, footnotes, endnotes, macro procedure text, annotation text, and text within textboxes are kept in separate subdocuments. Each subdocument has its own CP coordinate space. In other words, data structures are stored in Word files that are components of these subdocuments. These data

structures contain CP coordinates whose 0 point is the beginning of the subdocument text stream instead of the beginning of the main document text stream.

In **full-saved documents**, a simple calculation with values stored in the FIB produces the file offset of the beginning of the subdocument text streams (if they exist). The length of these streams is also stored.

In **fast-saved documents**, the **piece tables** of subdocuments are concatenated to the end of the main document piece table. In this case, to identify the beginning of subdocument text, you must sum the length of the main document text stream with the lengths of any subdocument text streams stored ahead of the subdocument (information stored in the FIB) and treat this sum as a CP coordinate. To retrieve the text of the subdocument, you must do lookups in the piece table, starting with the piece that contains the beginning CP coordinate, to find the physical location of each piece of the subdocument text stream.

summary information stream:
The stream within a Word .doc file containing the document summary information.

table row:
A contiguous sequence of paragraphs within the text stream of a document that is partitioned into subsequences of paragraphs called **cells**. The last paragraph of each cell is terminated by a special paragraph mark called a **cell mark**. Following the cell mark that ends the last cell of a table row, the table row is terminated by a special paragraph mark called a **row mark**. When Word displays a table row, it assigns a rectangular shaped display area to each cell in the row. All of the cell display area's tops are aligned at the same vertical position on a page. The leftmost display area in a table row is assigned to the 0th cell of the row; the next display area to the right is assigned to the 1st cell of the row, etc. The text of the cell is wrapped to fit its display area. As more text is added to the cell, the cell display area extends downward. A set of table properties that determine how many cells are in a row, where the horizontal boundaries of cell display areas are, and what borders are drawn around each cell in the table is stored for the **row mark** that marks the end of the table row.

table stream:
The stream within a Word .doc file containing the various plcf's and tables that describe a document's structures.

TAP (TAble Properties):
The data structure which describes the properties of a single table row. The information in the TAP for a table row is stored in a Word file as a list of sprms that modify a TAP which has been cleared to zeros. This list of table sprms is appended to the grpprl of paragraph sprms that is recorded in the PAPX for the **row mark** that delimits the end of a **table row**.

UPE (Universal Property Expansion)
Describes the "end-result" of property formatting, i.e. what the style looks like. The UPE structure is a non-zero prefix of a UPD structure.

UPX (Universal Property eXception)
Describes the difference in formatting of a style as compared to its based-on style.

XCHAR(eXtended CHARacter set):
A data type which defines a "character". Each XCHAR corresponds to a character in the document, where "character" is defined as a glyph, regardless of whether it is a single-byte or double-byte character. With Word6 (East Asian), Word95 (East Asian), Word97/all and future versions of Word, this is defined as a 16-bit integer corresponding to the Unicode character code of the glyph.

XST

Note In this document, bit 0 is the low-order bit. Structures are described as they would be declared in C for the Intel architecture. When numbering bytes in a word from low offset towards high offset, two-byte integers have their least significant eight bits stored in byte 0 and most significant eight bits in byte 1. If bit 31 is the most significant bit in a four-byte integer, bits 31 through 24 are stored in byte 3 of a four-byte integer, bits 23 through 16 are stored in byte 2, bits 15 through 8 will be stored in byte 1, and bits 7 through 0 are stored in byte 0.

Naming Conventions

The field names in Word data structures usually consist of a prefix of lower case characters followed by an optional upper case modifier. The following tags are used in the lower case prefix of field names to document the data type of the field:

`b` Used to name a 1 byte integer value

`c` Prefix used to signify that an integer value is a count of some number of objects. (e.g. a `cb` is a count of bytes, a `cl` is a count of lines, `ccol` is a count of columns, a `cpe` is a count of picture elements.)

`cp` Used to name a variable that contains a character position within the document. Always a 4 byte quantity.

`dxa` Used to name a variable that contains the horizontal distance of an object measured from some reference point expressed in twips. (e.g. `pap.dxaLeft` is the distance of the left boundary of a paragraph measured from the left margin of the page). See "xa" for definition of twip.

`dxp` Used to name a variable that contains the horizontal distance of an object measured from some reference point expressed in Macintosh pixel units (1/72"). (e.g. `dxpSpace`)

`dya` Used to name a variable that contains the vertical distance of an object measured from some reference point expressed in twips. (e.g. `pap.dyaAbs` is the vertical distance of the top of a paragraph from a reference frame declared in the `pap`). See "xa" for definition of twip.

`dyp` Used to name a variable that contains the vertical distance of an object measured from some reference point expressed in Macintosh pixel units (1/72").

`f` Used to name a flag (a variable containing a Boolean value). Usually the object referred to will contain either 1 (`fTrue`, `TRUE`) or 0 (`fFalse`, `FALSE`). (e.g. `fWidowControl`, `fShadow`)

`fc` Used to name a variable that contains an offset from the beginning of a file. Always a 4 byte quantity.

`grp` Prefix used to name an array of bytes that contains one or more copies of a variable length data structure with the instances of the data structure stored one after the other in the array. (e.g. a `grpprl` is an array of bytes that stores a group of `prl`s.)

`grpf` Prefix used to name an integer or byte value whose bits are used as flags. (e.g. `grpfIhdt` is a group of flags that records the types of headers that are stored for a particular section of a document).

`i` Prefix used to signify that an integer value is used as an index into an array. (e.g. `itbd` is an index into `rgtbd`, `itc` is an index into `rgtc`.)

`l` Used to name a 4 byte integer value (a long). (e.g. `lcb`)

`rg` Prefix used to signify that the data structure being defined is an array. (e.g. `rgb` (an array of bytes), `rgcp` (an array of `CP`s), `rgfc` (an array of `FC`s), `rgfoo` (an array of `foo`s).

`w` Used to name a 2 byte integer value (a short).

xa Used to name a variable that contains a width of an object imaged on screen or on hard copy that is measured in units of 1/1440 of an inch. This unit which is one-twentieth of a point size (1/20 * 1/72") is called a **twip** in this documentation. (e.g. `xaPage` is the width of a page).

ya Used to name a variable that contains the height of an object imaged on screen or on hard copy that is measured in twips. See "xa" for definition of twip.

The two following modifiers are used occasionally in this documentation:

First Means that the variable marks the first of a range of objects. For example, `cpFirst` would mark the first character position of a range of characters in a document. `fcFirst` would mark the file offset of the first byte of a range of bytes stored in a file.

Lim Means the variable marks the limit of a range of objects (i.e. is the index of the last object in a range plus 1). For example, `cpLim` would be the limit `CP` of a range of characters in a document. `fcLim` would be the limit file offset of a range of bytes stored in a file.

Format of the Summary Info Stream in a Word File

The summary information for a Word document is stored in two structured storage streams, `SummaryInformation` and `DocumentSummaryInformation`.

SummaryInformation and DocumentSummaryInformation are widely understood. You can find additional information at:

- http://msdn2.microsoft.com/en-us/library/aa380376(VS.85).aspx
- http://poi.apache.org/apidocs/org/apache/poi/hpsf/SummaryInformation.html
- http://poi.apache.org/apidocs/org/apache/poi/hpsf/DocumentSummaryInformation.html

.

Format of the Main Stream in a Word Non-Complex File

The main stream of a Word .doc file (non-complex format) consists of the Word file header (`FIB`), the text, and the formatting information.

FIB
Stored at the beginning of page 0 of the file. `fib.fComplex` is set to zero.

Text of body, footnotes, headers
Text begins at the position recorded in `fib.fcMin`.

FKPs for CHPs, PAPs and LVCs
The first Formatted Disk Page (`FKP`) begins at a 512-byte boundary after the last byte of text is written. The remaining `FKP`s are sequentially recorded in the 512-byte pages following the first `FKP`. The `FKP`s for Character Properties (`CHP`s), Paragraph Properties (`PAP`s), and `LVC`s are interleaved. Previous versions of Word wrote them in contiguous blocks. The `hplcfbte`'s of the three flavors (`CHP`, `PAP` and `LVC`) are used to find the relevant `FKP` of the appropriate type.

Group of SEPXs
Section Property Exceptions (SEPXs) immediately follow the FKPs and are concatenated one after the other. SEPXs are no longer guaranteed to start on a page boundary if it would span a boundary when placed immediately after the preceding SEPX.

Format of the Main Stream in a Complex File

The main stream of a Word binary file (complex format) consists of the Word file header (FIB), the text, and the formatting information.

FIB
Stored at beginning of page 0 of the file. fib.fComplex is set to one.

Text of body, footnotes, headers stored during last full save
Text begins at the position recorded in fib.fcMin.

FKPs for CHPs, PAPs and LVCs
The first Formatted Disk Page (FKP) begins at a 512-byte boundary after the last byte of text is written. The remaining FKPs are sequentially recorded in the 512-byte pages following the first FKP. The FKPs for Character Properties (CHPs), Paragraph Properties (PAPs), and LVCs are interleaved. Previous versions of Word wrote them in contiguous blocks. The hplcfbte's of the three flavors (CHP, PAP and LVC) are used to find the relevant FKP of the appropriate type.

Group of SEPXs stored during last full save
SEPXs immediately follow the FKPs and are concatenated one after the other.

Any text, stored during first fast save
Any FKPs stored during first fast save
Any SEPXs stored during first fast save

Any text, stored during second fast save
Any FKPs stored during second fast save
Any SEPXs stored during second fast save

...

Any text, stored during nth fast save
Any FKPs stored during nth fast save
Any SEPXs stored during nth fast save

Format of the Table Stream

Word stores various plcfs and tables with the stream named either "0Table" or "1Table". Ordinarily a file will contain only one table stream. However, in some unusual circumstances (e.g. crash during file save) a file might have two table streams. In that case the bit field fWhichTblStm in the FIB should be used to determine which table stream to read. If fWhichTblStm==0, then the FIB refers to the stream named "0Table", and if fWhichTblStm==1, then the FIB refers to the stream name "1Table".

autosaveSource (name of original)
Written immediately after the sttbfAssoc table. This field only appears in auto saved files. These files are normal Word documents in every other way. Also, auto saved files are typically in the complex file format except that the tables (plcf*, etc.) are not overwritten. For example, an auto saved file is typically longer than the equivalent Word document.

bkdEdn (endnote text break descriptor table)
Written immediately after the pgdEdn if the document contains endnotes

bkdFtn (footnote text break descriptor table)
Written immediately after the pgdFtn if the document contains footnotes.

bkdMother (break descriptor table)
Written immediately after the pgdMother in all Word documents

clx (encoding of the sprm lists for a complex file and piece table for a any file)
Written immediately after the end of the previously recorded structure. This is recorded in all Word documents.

cmds (recording of command data structures)
Written immediately after the previously recorded table, if special commands are linked to this document.

dggInfo (Office drawing information)
Written immediately after the previously recorded table. Format is described in the Office drawing group format document.

dop (document properties record)
Written immediately after the end of the previously recorded structure. This is recorded in all Word documents

formFldSttbs (form field dropdown string tables)
Written immediately after the previously recorded table, if the document contains form field dropdown controls.

grpXstAtnOwners (annotation owner table)
Written immediately after the previously recorded table if the document contains annotations.

hplgosl (grammar option settings)
Written immediately after the previously recorded table. This undocumented structure maps LID and grammar checker type to grammar checking options.

pgdEdn (endnote text page description table)
Written immediately after the plcfendTxt if the document contains endnotes

pgdFtn (footnote text page description table)
Written immediately after the plcffndTxt if the document contains footnotes

pgdMother (page description table)
Written immediately after the plcfsed in all Word documents

plcasumy (AutoSummary analysis)
Written immediately after the previously recorded table, if the document stored is in AutoSummary view mode.

plcfandRef (annotation reference position table)
Written immediately after the grpXstAtnOwners if the document contains annotations

plcfandTxt (annotation text position table)
Written immediately after the plcfandRef if the document contains annotations.

plcfAtnbkf (table recording the beginning CPs for bookmarks in the annotation subdocument)
Written immediately after the sttbfAtnBkmk previously recorded table, if the document contains annotations with bookmarks.

plcfAtnbkl (table recording the limit CPs of bookmarks in the annotation subdocument)
Written immediately after the plcfAtnbkf previously recorded table, if the document contains annotations with bookmarks.

plcfBkmkf (table recording beginning CPs of bookmarks)
Written immediately after the `sttbfBkmk`, if the document contains bookmarks.

plcfBkmkl (table recording limit CPs of bookmarks)
Written immediately after the `plcfBkmkf`, if the document contains bookmarks.

plcfbteChpx (bin table for CHP FKPs)
Written immediately after the previously recorded table. This is recorded in all Word documents.

plcfbtePapx (bin table for PAP FKPs)
Written immediately after the `plcfbteChpx`. This is recorded in all Word documents.

plcfbteLvc (bin table for LVC FKPs)
Written immediately after the `plcfbtePapx`. This is recorded in all Word documents.

plcfendRef (endnote reference position table)
Written immediately after the previously recorded table if the document contains endnotes

plcfendTxt (endnote text position table)
Written immediately after the `plcfendRef` if the document contains endnotes

plcffldAtn (table of field positions and statuses for annotation subdocument)
Written immediately after the previously recorded table, if the annotation subdocument contains fields.

plcffldEdn (table of field positions and statuses for endnote subdocument)
Written immediately after the previously recorded table, if the endnote subdocument contains fields.

plcffldFtn (table of field positions and statuses for footnote subdocument)
Written immediately after the previously recorded table, if the footnote subdocument contains fields.

plcffldHdr (table of field positions and statuses for header subdocument)
Written immediately after the previously recorded table, if the header subdocument contains fields.

plcffldHdrTxbx (table of field positions and statuses for textbox subdocument of header subdocument)
Written immediately after the previously recorded table, if the textbox subdocument of the header subdocument contains fields.

plcffldMom (table of field positions and statuses for main document)
Written immediately after the previously recorded table if the main document contains fields.

plcffldTxbx (table of field positions and statuses for textbox subdocument)
Written immediately after the previously recorded table, if the textbox subdocument contains fields.

plcffndRef (footnote reference position table)
Written immediately after the `stsh` if the document contains footnotes

plcffndTxt (footnote text position table)
Written immediately after the `plcffndRef` if the document contains footnotes

plcfglsy (glossary entry text position table)
Written immediately after the previously recorded table, if the document stored is a glossary.

plcfhdd (header text position table)
Written immediately after the previously recorded table, if the document contains headers or footers.

plcfhdrtxbxBkd (header text box break descriptor table)
Written immediately after the `plcfhdrtxbxTxt` if the header subdocument contains textboxes.

plcfhdrtxbxTxt (header text box link table)
Written immediately after the previously recorded table if the header subdocument contains textboxes

plcfgram (grammar state table)
Written immediately after the previously recorded table. Records state of grammar checking in a `PLCF` of `SPLS` structures.

plcflst (list formats)
Written immediately after the end of the previously recorded, if there are any lists defined in the document. This begins with a short count of `LSTF` structures followed by those `LSTF` structures.

This is immediately followed by the allocated data hanging off the `LSTF`s. This data consists of the array of `LVL`s for each `LSTF`. (Each `LVL` consists of an `LVLF` followed by two `grpprl`s and an `XST`.)

plcflvc (list and outline level table)
Written immediately after the previously recorded table during fast save only.

plcfphe (paragraph height table)
Written after the previously recorded table, if paragraph heights were recorded. Only written during a fast save.

plcfsea (private)
`PLCF` reserved for private use by Word.

plcfsed (section table)
Written immediately after the previously recorded table. Recorded in all Word documents

plcfspl (spelling state table)
Written immediately after the previously recorded table. Records state of spell checking in a `PLCF` of `SPLS` structures.

plcftxbxBkd (text box break descriptor table)
Written immediately after the `plcftxbxTxt` if the document contains textboxes

plcftxbxTxt (text box link table)
Written immediately after the previously recorded table if the document contains textboxes

plcOcx (ocx position table)
Written immediately after the previously recorded table, if the document contains OLE controls. Undocumented.

plcspaHdr (header Office drawing table)
Written immediately after the previously recorded table, if the header subdocument contains Office drawings.

plcspaMom (Office drawing table)
Written immediately after the previously recorded table, if the document contains Office drawings.

plcupcRgbuse
Undocumented undo / versioning data

plcupcUsp
Undocumented undo / versioning data

plcfwkb (work book document partition table)
Written immediately after the previously recorded table, if the document is a master document.

plflfo (more list formats)
Written immediately after the end of the `plcflst` and its accompanying data, if there are any lists defined in the document. This consists first of a `PL` of `LFO` records, followed by the allocated data (if any) hanging off the `LFO`s. The allocated data consists of the array of `LFOLVLF`s for each `LFO` (and each `LFOLVLF` is immediately followed by some `LVL`s).

pms (print merge state)
Written immediately after the previously recorded table, if information about the print / mail merge state is recorded for the document

prDrvr (printer driver information)
Written immediately after the previously recorded table, if a print environment is recorded for the document.

prEnvLand (print environment in landscape mode)
Written immediately after the previously recorded table, if a landscape mode print environment is recorded for this document.

prEnvPort (print environment in portrait mode)
Written immediately after the previously recorded table, if a portrait mode print environment is recorded for this document.

routeSlip (mailer routing slip)
Written immediately after the previously recorded table, if this document has a mailer routing slip.

sttbAutoCaption (auto caption string table)
Written immediately after the previously recorded table, if the document contains auto captions.

sttbCaption (caption title string table)
Written immediately after the previously recorded table, if the document contains captions.

sttbfAssoc (table of associated strings)
Table of associated strings.

sttbfAtnBkmk (table of annotation bookmark string names)
Written immediately after the previously recorded table, if the document contains annotations with bookmarks.

sttbfBkmk (table of bookmark name strings)
Written immediately after the previously recorded table, if the document contains bookmarks.

sttbFnm (filename reference string table)
Written immediately after the previously recorded table, if the document references other documents.

sttbfListNames (more list formats)
Written immediately after the end of the `plflfo` and its accompanying data, if there are any lists defined in the document. This is a string table containing the list names for each list. It is parallel with the `plcflst`, and may contain null strings if the corresponding `LST` does not have a list name.

sttbfffn (table of font name strings)
Written immediately after the `clx`. This is recorded in all Word documents. The `sttbfffn` is an `sttbf` where each string is instead an `FFN` structure (note that just as for a Pascal-style string, the first byte in the `FFN` records the total number of bytes not counting the count byte itself). The names of the fonts correspond to the `ftc` codes in the `CHP` structure. For example, the first font name listed corresponds is the name for `ftc=0`.

sttbfRMark (revision mark author string table)
Written immediately after `plcfbteLvc`, if the document contains revision marks.

sttbfUssr
Undocumented undo / versioning data

sttbGlsy (glossary name string table)
Written immediately after the previously recorded table, if the document stored is a glossary.

sttbGlsyStyle (glossary style name string table)
Written immediately after `sttbGlsy`, if the document stored is a glossary.

sttbSavedBy (last saved by string table)
Written immediately after the previously recorded table.

sttbttmbd (true type font embedding string table)
Written immediately after the end of the previously recorded structure if the document contains embedded true type fonts.

stsh (style sheet)
Written immediately after the previous table. This is recorded in all Word documents.

stwUser (macro user storage)
Macro user storage.

uskf
Undocumented undo / versioning data

wss (window state structure)
Written immediately after the end of previously recorded structure, if the document was saved while a window was open.

Format of the Data Stream

embedded objects-native data
Word embedded object structures are sequentially concatenated if the document contains embedded objects.

huge PAPXs
The `grpprl`s from `PAPX`s which are too large to fit in an `FKP` are sequentially concatenated as necessary.

pictures
Word picture structures are sequentially concatenated if the document contains pictures.

Format of the Custom XML Storage (Added in Word 2007)

This storage specifies the custom XML parts inside of a binary format for Microsoft Word 97. For additional information on custom XML parts, see the Office Open XML specification section on the element “datastoreItem”.

In the binary format for Microsoft Word 97, the custom XML parts are stored inside a storage called “MSODatastore”. Within this fstorage, zero or more custom XML parts can exist each in their own storage. Each of these storages is stamped with a unique identifier as its storage name. An instance of one of these storages contains two streams within it:

1. A stream named item
2. A stream named properties

For information on these two streams, see the Office Open XML specification section on “Custom XML Data Storage Part” and “Custom XML Data Storage Properties Part”.

FIB

The `FIB` contains a "magic word" and pointers to the various other parts of the file, as well as information about the length of the file. The `FIB` starts at the beginning of the file. The `FIB` is defined in the structure definition section of this document.

Text

The text of the file starts at `fib.fcMin` and is usually set to the next 128 byte boundary after the end of the `FIB`. The text in a Word document is ASCII text with the following restrictions (ASCII codes given in decimal):

- **Paragraph ends** are stored as ASCII 13 (a single <Carriage Return> character). No other occurrences of this character sequence are allowed.
- **Hard line breaks** which are not paragraph ends are stored as ASCII 11. Other line break or word wrap information is not stored.

  Hyphens

  - **Breaking hyphens** are stored as ASCII 45 (normal hyphen code).
  - **Non-required hyphens** are ASCII 31.
  - **Non-breaking hyphens** are stored as ASCII 30.
- **Non-breaking spaces** are stored as 160.
- Normal **spaces** are ASCII 32.
- **Page breaks** and **Section marks** are ASCII 12 (normal form feed); if there's an entry in the section table, it's a section mark, otherwise it's a page break.
- **Column breaks** are stored as ASCII 14.
- **Tab** characters are ASCII 9 (normal).

  Fields

  - **Field begin mark** which delimits the beginning of a field is ASCII 19.
  - **Field end mark** which delimits the end of a field is ASCII 21.
  - **Field separator** ,which marks the boundary between the preceding field code text and following field expansion text within a field, is ASCII 20.
  - **Field escape character** is the '\' character which also serves as the **formula mark**.
- The **cell mark** which delimits the end of a cell in a table row is stored as ASCII 7 and has the `fInTable` paragraph property set to `fTrue (pap.fInTable==1)`.
- The **row mark** which delimits the end of a table row is stored as ASCII 7 and has the `fInTable` paragraph property and `fTtp` paragraph property set to `fTrue (pap.fInTable==1 && pap.fTtp==1)`.

The following ASCII codes are treated as "special" characters when they have the character property *special* on (`chp.fSpec==1`):

ASCII code	Special character
0	Current page number
1	Picture
2	Auto numbered footnote reference.
3	Footnote separator character
4	Footnote continuation character
5	Annotation reference
6	Line number
7	Hand Annotation picture (Generated in Pen Windows)

ASCII code	Special character
8	Drawn object
10	Abbreviated date (e.g. "Wed, Dec 1, 1993")
11	Time in hours:minutes:seconds
12	Current section number
14	Abbreviated day of week (e.g. "Thu" for "Thursday")
15	Day of week (e.g. "Thursday")
16	Day short (e.g. "9" for the ninth day of the month)
22	Hour of current time with no leading zero
23	Hour of current time (two digit with leading zero when necessary)
24	Minute of current time with no leading zero
25	Minute of current time(two digit with leading zero when necessary)
26	Seconds of current time
27	AM/PM for current time
28	Current time in hours:minutes:seconds in old format
29	Date M (e.g. "December 2, 1993")
30	Short Date (e.g. "12/2/93")
33	Short Month (e.g. "12" to represent "December")
34	Long Year (e.g. "1993")
35	Short Year (e.g. "93")
36	Abbreviated month (e.g. "Dec" to represent "December")
37	Long month (e.g. "December")
38	Current time in hours:minutes (e.g. "2:01")
39	Long date (e.g. "Thursday, December 2, 1993")
41	Print Merge Helper field

Note The end of a section is also the end of a paragraph. The last character of a section is a section mark which stands in place of the paragraph mark normally required to end a paragraph. An exception is made for the last character of a document which is always a paragraph mark although the end of a document is always an implicit end of section.

If `!fib.fComplex`, the document text stream is represented by the text beginning at `fib.fcMin` up to (but not including) `fib.fcMac`. Otherwise, the document is represented by the piece table stored in the file in the data beginning at `fib.fcClx`.

The document text stream includes text that is part of the main document, plus any text that exists for the footnote, header, macro, or annotation subdocuments. The sizes of the main document and the header, footnote, macro and annotation subdocuments are stored in the `fib`, in variables:

`fib.ccpText`, `fib.ccpFtn` `fib.ccpHdr`

`fib.ccpMcr` `fib.ccpEdn` `fib.ccpTxbx`

`fib.ccpHdrTxbox` `fib.ccpAtn`

In a non-complex file, this means text of the:

- main document begins at `fib.fcMin` in the file and continues through `fib.fcMin+fib.ccpText`.
- footnote subdocument begins at `fib.fcMin+fib.ccpText` and extends to `fib.fcMin+fib.ccpText+fib.ccpFtn`.
- header subdocument begins at `fib.fcMin+fib.ccpText+fib.ccpFtn` and extends to `fib.fcMin+fib.ccpText+fib.ccpFtn+fib.ccpHdr`.
- annotation subdocument begins at `fib.fcMin+fib.ccpText+fib.ccpFtn+fib.ccpHdr` and extends to `fib.fcMin+fib.ccpText+fib.ccpFtn+fib.ccpHdr+ccpAtn`.
- endnote subdocument begins at `fib.fcMin+fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpAtn` and extends to `fib.fcMin+fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpEdn`.
- textbox subdocument begins at `fib.fcMin+fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpAtn+fib.ccpEdn` and extends to `fib.fcMin+fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpEdn+fib.ccpTxbx`.
- header textbox subdocument begins at `fib.fcMin+fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpAtn+fib.ccpEdn+fib.ccpTxbx` and extends to `fib.fcMin+fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpEdn+fib.ccpTxbx+fib.ccpHdrTxbx`.

In a complex, fast-saved file text of the:

- main document must be found by examining the piece table entries from the 0th piece table entry from the piece table entry that describes `cp=fib.ccpText`.
- footnote subdocument must be found by examining the piece table entries, beginning with the one that describes `cp=fib.ccpText` through the entry that describes `cp=fib.ccpText+fib.ccpFtn`.
- header subdocument must be found by examining the piece table entries, beginning with the one that describes `cp=fib.ccpText+fib.ccpFtn` through the entry that describes `cp=fib.ccpText+fib.ccpFtn+fib.ccpHdr`.
- annotation subdocument must be found by examining the piece table entries, beginning with the one that describes `cp=fib.ccpText+fib.ccpFtn+fib.ccpHdr` through the entry that describes `cp=fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpAtn`.
- endnote subdocument must be found by examining the piece table entries beginning with the one that describes `cp=fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpAtn` through the entry that describes `cp=fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpAtn+fib.ccpEdn`.
- textbox subdocument must be found by examining the piece table entries beginning with the one that describes `cp=fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpAtn+fib.ccpEdn` through the entry that describes `cp=fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpAtn+fib.ccpEdn+fib.ccpTxbx`.
- header textbox subdocument must be found by examining the piece table entries beginning with the one that describes `cp=fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpAtn+fib.ccpEdn+`

fib.ccpTxbx through the entry that describes cp=fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpAtn+fib.ccpEdn+fib.ccpTxbx+fib.ccpHdrTxbx.

Character and Paragraph Formatting Properties

Character and paragraph properties in Word documents are stored in a compressed format. The information stored on disk is not the properties of a particular sequence of text but the difference of the properties from a specific reference property.

The PAP is a data structure that holds uncompressed paragraph property information; the CHP (pronounced "chip") is a structure that holds uncompressed character property information. Each paragraph in a Word document inherits a default set of paragraph and character properties from one of the **paragraph styles** recorded in the style sheet data structure (STSH).

A particular PAP is converted into its compressed form, the PAPX, by first comparing the pap for a paragraph with the pap stored in the style sheet for the paragraph's style. Any properties in the paragraph's PAP that are different from those stored in the style sheet PAP are encoded as a list of sprms (grpprl). sprms express how the content of the style sheet PAP should be transformed to create the properties for the paragraph. A PAPX is a variable-length data structure that begins with a count of words that encodes the PAPX length. It contains an istd (index to style descriptor) which specifies which style entry in the style sheet contains the default paragraph and character properties for the paragraph, paragraph height information, and the list of difference sprms. If the only difference between the paragraph's PAP and the style's PAP were in the justification code field, which is one byte long, one two-byte sprm, sprmPJc, would be generated to express that difference; thus the total PAPX size would be 5 bytes. This is better than 54-1 compression since the total size of a PAP is 274 bytes.

To convert a CHP for a sequence of characters contained within a single paragraph into its compressed form, the CHPX, it's first necessary to know the **paragraph style** assigned to the paragraph containing those characters and any character style that may be tagging the character run. The character properties inherited from the paragraph style are moved into a buffer. If the chp.istd of the chp to be compressed is not istdNormalChar, the changes recorded for that character style are applied to the buffer. Then the character properties of the character sequence are compared with the character properties generated using the paragraph's style and the run’s character style. Any properties in the paragraph's CHP that are different from those stored in the generated CHP are encoded as a list of sprms (grpprl). The sprms express how the content of the CHP generated from the paragraph and character styles should be transformed to create the character properties for the text run. A CHPX is a variable-length data structure that begins with a count of words that encodes the CHPX length followed by the list of difference sprms.

If one of the bit fields in the CHP to be compressed such as fBold is different from the reference CHP, you would build a difference sprm using sprmCFBold in the first byte and the bytes pattern 0x81 in the second byte which signifies that the value of the bit in the CHP to be compressed is of opposite value from the value stored in the reference CHP. If there was no difference, sprmCFBold would not be recorded in the grrprl to be generated. If there were a difference in a field larger than a single bit such as the chp.hps, a sprmCHps would be generated to record the value of chp.hps in the chp to be compressed. If the chp.hps were equal in both the chp to be compressed and the reference CHP, sprmCHps would not be recorded in the grrprl that is generated. If a sequence of characters has the same character properties and the sequence spans more than one paragraph, it's necessary to examine each paragraph's properties and to generate a different CHPX every time there is a change of style.

In Word documents, the fundamental unit of text for which character exception information is kept is the **run of exception text**, a contiguous sequence of characters stored on disk that all have the same exception properties with respect to their underlying style character properties. Each run would have an entry recorded in a CHPX FKP. If a user never changed the character properties inherited from the styles used in the document and did a complete save of the document, although each of those styles may have different properties, the entire document stream would be one large **run of exception text** and one CHPX would suffice to describe the character properties of the entire document.

The fundamental unit of text for which paragraph properties are recorded is the **paragraph**. Every paragraph has an entry recorded in a PAPX FKP.

The CHPX FKP and the PAPX FKP have similar physical structures. An FKP is a 512-byte data structure that is stored in one page of a Word file. At offset 511 is a 1-byte count named crun, which is a count of runs of exception text for CHPX FKPs and which is a count of paragraphs in PAPX FKPs. Beginning at offset 0 of the FKP is an array of crun+1 FCs, named rgfc, which records the beginning and limit FCs of crun runs of exception text or paragraphs.

For CHPX FKPs, immediately following fkp.rgfc is a byte array of crun word offsets to CHPXs from the beginning of the FKP. This byte array, named rgb, is in 1-to-1 correspondence with the rgfc.

For PAPX FKPSs, immediately following the fkp.rgfc is an array of 13 byte entries called BXs. This array called the rgbx is in 1-to-1 correspondence with the rgfc. The first byte of the **ith** BX entry contains a single byte field which gives the word offset of the PAPX that belongs to the paragraph whose beginning in FC space is rgfc[i] and whose limit is rgfc[i+1] in FC space. The last 12 bytes of the **ith** BX entry contain a PHE structure that stores the current paragraph height of the paragraph whose beginning in FC space is rgfc[i] and whose limit is rgfc[i+1] in FC space.

The fact that the offset to properties stored in the rgb or rgbx is a word offset implies that CHPXs and PAPXs are stored in FKPs beginning on word boundaries. Since the values stored in the rgb/rgbx allow random access throughout the FKP, space within an FKP can be conserved by storing the offset of the same physical CHPX/PAPX in rgb/rgbx entries when several runs or paragraphs in the FKP have the same properties. Word uses this optimization.

An rgb or rgbx[].b value of 0 is used in another optimization. When an rgb or rgbx[].b value of 0 is stored in an FKP, it means that instead of referring to a particular CHPX/PAPX in the FKP the 0 value signals the reader to construct a commonly encountered predefined set of properties.

For CHPX FKPs a 0 rgb value means the properties of the run of text were exactly equal to the character properties inherited from the style of the paragraph it was in. For PAPX FKPs, a 0 rgbx[].b value means the paragraph's properties were exactly equal to the paragraph properties of the Normal style (stc==0) and the paragraph contained 1 line of 240 pixels, with a column width of 7980 dxas.

When new entries are added to an FKP, there must be unallocated space in the middle of the FKP equal to 5 bytes for CHPXs (size of an FC plus size of one-byte word offset) or 11 bytes for PAPXs (size of an FC plus the size of a seven byte BX entry), plus the size of the new CHPX or PAPX if the property being added is not already recorded in the FKP and is not the property coded with a 0 rgb/rgbx[].b value. To add a new property in a CHPX FKP, existing rgb entries are moved four bytes to the right in the FKP. To add a new property in a PAPX FKP, existing rgbx entries are moved four bytes to the right in the FKP. The new FC is added at the end of the rgfc. The new CHPX or PAPX is recorded on a 2-byte boundary before the previously recorded properties are stored at the end of the block. The word offset of the beginning of the CHPX or PAPX is stored as the last entry of the relocated rgb/rgbx[].b, and finally, the crun stored at offset 511 is incremented. In Word '97, PAPXs can be generated

which are too large to fit in an FKP. In such a case, the grpprl of the PAPX is written to the data stream and a PAPX is stored in an FKP with that grpprl replaced by a sprmPHugePapx.

Bin Tables

A bin table (plcfbte) partitions the total extent of the Word file that contains text characters into a set of contiguous intervals marked by an fcFirst and an fcLim. The fcFirst for the **nth** interval would be plcfbte.rgfc[n] and the fcLim for the **nth** interval would be plcfbte.rgfc[n+1]. Associated with each interval is a BTE. A BTE holds a four-byte PN (page number) which identifies the FKP page in the file which contains the formatting information for that interval. A CHPX FKP further partitions an interval into runs of exception text. A PAPX FKP in a non-complex, full-saved file, partitions the text within intervals into paragraphs. If a file is in complex format (was fast-saved), the PAPX FKP only records the FCs within the text preceded by a paragraph mark. Even though a sequence of text may be between two paragraph end marks, it may reside in a paragraph different from the one defined by the next paragraph end mark, because the text may have been moved by the user into a different paragraph. In the logical text stream represented by the document's piece table, the paragraph mark that follows the moved text is stored in a non-adjacent physical location in the file.

Style Sheet

A style sheet is a collection of styles. In Word, each document has its own style sheet.

A style is a collection of formatting information with a name. Word 6.0 and later versions support paragraph and character styles. Versions of Word prior to 6.0 support only paragraph styles. Character styles have just character formatting. Paragraph styles have both character and paragraph formatting. The style sheet establishes a correspondence between a style code and a style definition.

Note: the storage and behavior of styles has changed considerably since WinWord 2.x, beginning with nFib 63. Some of the differences are:

- Character styles are supported.
- The style code is called an istd, rather than an stc.
- The istd is a short integer, where the stc was a byte.
- The range of the istd is 0-4095, where 4095 is the null style. The range of the stc was 0-256, with 222 as the null style.
- PAPX's have a short istd at the beginning, rather than a byte stc.
- CHPX's are a grpprl, not a CHP.
- Many other changes...

The styles for a document (both paragraph and character styles) are stored in an array in each document. [The DOD.hplhqstd is a handle to a plex (array) of hq's (handles) to std's (style descriptions] When new styles are created, they are added to the end of the array. The array can have unused slots. Some slots at the beginning of the array are reserved for specific styles, whether they were created yet or not. [Istd (slot) 0 is Normal. Istd 1-9 are Heading 1-9. Istd 10 is Default Paragraph Font. Istd 11-14 are reserved. So the first non-fixed index is 15 (see stshi.istdMaxFixedWhenSaved.] Paragraph and character styles are stored in the same array. Each document has a separate array, so the same style will usually [Those styles in fixed locations in the style sheet will have the same istd's in all documents] have a different istd in two different documents. Thus style matching between documents must be done by name (or by sti if the styles are built-in).

Styles are usually referred to using an istd. The istd is an index into an array of STDs (STyle Descriptions). A (doc, istd) pair uniquely identifies a style because it tells which style is in which array.

Parts of a style (for more information, see the STD structure below):

- sti: A style identifier. Built-in styles have a unique sti to indicate which built-in style they reference. User-defined styles use stiUser.
- stk: The type of style, either paragraph or character.
- istdBase: The style that this style is based on.
- istdNext: The style that should be applied after this one.
- stzName: The name of a style, unique within its style sheet.
- UPX: The difference between this style and the one it is based on.
- UPE: The properties of this style (a PAP, CHP, and/or grpprl).

Every paragraph has a paragraph style. Every character has a character style. The default paragraph style is Normal (stiNormal, istdNormal). The default character style is Default Paragraph Font (stiNormalChar, istdNormalChar).

The formatting of a paragraph (the PAP) and a character (the CHP) depend on the paragraph and character styles applied to them, as well as any additional formatting stored in the FKPs. The PAP and CHP are constructed in a layered fashion:

For a PAP:

1. An initial PAP is determined by getting the PAP from the paragraph's style.
2. Any paragraph formatting stored in the file (the FKP PAPX's) is then applied to that PAP.

For a CHP:

1. An initial CHP is determined by getting the CHP from the paragraph's style.
2. Properties from the character's style (the UPX.chpx.grpprl) are then applied to that CHP.
3. Any character formatting stored in the file (the FKP CHPX's) is then applied to that CHP.

Note: the resulting PAP and CHP have fields that indicate what style was applied: PAP.istd, CHP.istd.

Stylesheet File Format

The style sheet (STSH) is stored in the file in two parts, a STSHI and then an array of STDs. The STSHI contains general information about the following style sheet, including how many styles are in it. After the STSHI, each style is written as an STD. Both the STSHI and each STD are preceded by a ushort that indicates their length.

Field	Size	Comment
cbStshi	2 bytes	Size of the following STSHI structure [1]
STSHI	(cbStshi)	Stylesheet Information

[1] For early versions of Word 6.0 files (versions prior to nFib 67), this field was not written. The cbStshi to use for those file versions is 4 bytes.

Then for each style in the style sheet (stshi.cstd), the following is stored:

cbStd	2 bytes	Size of the following STD structure

STD (cbStd) The style description

STSHI

The STSHI structure, which stores style sheet information has the following format:

```
typedef struct _STSHI
   {
   ushort   cstd; // Count of styles in stylesheet
   ushort   cbSTDBaseInFile;        // Length of STD Base as stored in a file
   BF    fStdStylenamesWritten : 1; // Are built-in stylenames stored?
   BF    : 15;                      // Spare flags
   ushort   stiMaxWhenSaved;        // Max sti known when this file was written
   ushort   istdMaxFixedWhenSaved;  // How many fixed-index istds are there?
   ushort   nVerBuiltInNamesWhenSaved; // Current version of built-in stylenames
   FTC      rgftcStandardChpStsh[iftcCompositeMax];   /* rgftc used by
                                 StandardChpStsh for this document */
   ushort  cbLSD;          /* size of each lsd in mpstilsd. The count of lsd's
                              is stiMaxWhenSaved */
   LSD      mpstilsd[stiMax]; /* latent style data
                                 (stiMax == stiMaxWhenSaved upon save!) */
   } STSHI;
```

The cb preceding the STSHI in the file is the length of the STSHI as stored in the file. The current definition of the STSHI structure might be longer or shorter than that stored in the file, the style sheet reader routine needs to take this into account.

stshi.cstd: The number of styles in this style sheet. There will be stshi.cstd (cbSTD, STD) pairs in the file following the STSHI. **Note**: styles can be empty, i.e. cbSTD==0.

stshi.cbSTDBaseInFile: The STD structure (see below) is divided into a fixed-length "base", and a variable length part. The stshi.cbSTDBaseInFile indicates the size in bytes of the fixed-length base of the STD as it was written in this file. If the STD base is grown in a future version, the file format doesn't change, because the style sheet reader can discard parts it doesn't know about, or use defaults if the file's STD is not as large as it was expecting. (Currently, stshi.cbSTDBaseInFile is 8.)

stshi.fStdStylenamesWritten: Previous versions of Word did not store the style name if the style was a built-in style; Word 6.0 stores the style name for compatibility with future versions. **Note**: the built-in style names may need to be "regenerated" if the file is opened in a different language or if stshi.nVerBuiltInNamesWhenSaved doesn't match the expected value.

stshi.stiMaxWhenSaved: This indicates the last built-in style known to the version of Word that saved this file.

stshi.istdMaxFixedWhenSaved: Each array of styles has some fixed-index styles at the beginning. This indicates the number of fixed-index positions reserved in the style sheet when it was saved.

stshi.nVerBuiltInNamesWhenSaved: Since built-in style names are saved with the document, this provides a way to see if the saved names are the same "version" as the names in the version of Word that is loading the file. If not, the built-in style names need to be "regenerated", i.e. the old names need to be replaced with the new.

stshi.rgftcStandardChpStsh: This is a list of the default fonts for this style sheet. The first is for ASCII characters (0-127), the second is for East Asian characters, and the third is the default font for non-East Asian, non-ASCII text. See notes on sprmCRgftcX for details.

Introduced in Word 2003:

stshi.cbLSD: This is the size of each LSD in mpstilsd. The count of LSD's is stiMaxWhenSaved.

stshi.mpstilsd[stiMax]. Latent style data (stiMax==stiMaxWhenSaved upon save!)

An array of LSD structures:

```
typedef struct _LSD
   {
   union
      {
      unsigned long grflsd;
      struct
         {
         BFL fLocked : 1;
         BFL : 31;
         };
       };
   } LSD;
```

fLocked indicates the style is currently locked, meaning it cannot be used in the document as a result of the Document Protection feature. The index into mpstilsd corresponds to the index of the style that the LSD structure affects (see std.sti below).

STD

Each individual style description is stored in an STD structure as follows:

```
typedef struct _STD
   {  // Base part of STD:
   ushort   sti : 12;      /* invariant style identifier */
   ushort   fScratch : 1;  /* spare field for any temporary use, always
                              reset back to zero! */
   ushort   fInvalHeight : 1;  /* PHEs of all text with this style are wrong */
   ushort   fHasUpe : 1;   /* UPEs have been generated */
   ushort   fMassCopy : 1; /* std has been mass-copied; if unused at save time,
                              style should be deleted */
   ushort   stk : 4;       /* style kind */
   ushort   istdBase : 12; /* base style */
   ushort   cupx : 4;      /* number of UPXs (and UPEs) */
   ushort   istdNext : 12; /* next style */
   ushort   bchUpe;        /* offset to end of upx's, start of upe's */
   ushort fAutoRedef : 1;  /* auto redefine style when appropriate */
   ushort fHidden: 1;      /* hidden from UI? */

   // These 2 flags are caches for the
   // information calc'd in Apply97LidsToIstd()
   ushort f97LidsSet: 1;   /* style already has valid sprmCRgLidX_80 in it */
   ushort fCopyLang: 1;    /* if f97LidsSet, says whether we copied the lid from
                              sprmCRgLidX into sprmCRgLidX_80 or whether we got
                              rid of sprmCRgLidX_80*/
   ushort fPersonalCompose: 1; /* HTML Threading compose style */
   ushort fPersonalReply: 1;   /* HTML Threading reply style */
   ushort fPersonal: 1;    /* HTML Threading - another user's personal style */
   ushort fNoHtmlExport:1; /* Do not export this style to HTML/CSS */
   ushort fSemiHidden: 1;  /* Do not show this style in long style lists */
   ushort fLocked:1;       /* locked style? */
   ushort fInternalUse:1;  /* Style is used by a word feature, e.g. footnote*/
   ushort : 5;             /* unused bits */
   ushort istdLink : 12;   /* is this style linked to another? */
#ifdef STYLERM
   ushort fHasOriginalStyle: 1; /* style has RevMarking history */
   ushort fSpare : 3;
#else
   ushort fSpare : 4;
#endif //STYLERM
   RSID  rsid;             /* marks during merge which doc's style changed */

   ushort iftcHtml : 3;    /* used temporarily during html export */
   ushort unused : 13;

   // Variable length part of STD:
   XCHAR xstzName[2];      /* sub-names are separated by chDelimStyle */
   /* char  grupx[]; */
   /* UPEs are not stored on the file; they are a cache of the based-on chain */
   /* char  grupe[]; */
   } STD;
```

The cb preceding each STD is the length of the data, which includes all of the STD except the grupe array (which is derived after the file is read in, by building each UPE from the base style

UPE plus the exceptions in the UPX.) A cb of zero indicates an empty slot in the style array, i.e. no style has that istd. **Note**: the STD structure may be longer or shorter than the one stored in the file; stshi.cbSTDBaseInFile indicates the length of the base of the STD (up to stzName) as stored in the file. The style sheet reader routine must take this into account.

The variable-length part of the STD has three variable-length subparts, the xstzName, the grupx, and the grupe. Since this doesn't fit well into a C structure declaration, some processing is needed to figure out where one part stops and the next part begins. An important note is that all variable-length parts and subparts of the STD begin on EVEN-BYTE OFFSETS within the STD, even if the length of the preceding variable-length part was odd.

std.sti: The sti is an identifier of which built-in style this is, or stiUser for a user-defined style. An sti is intended to be permanent throughout versions of Word, although new sti's may be added in new versions. The sti definitions are:

```
#define stiNormalPara  0   // 0x0000
#define stiHeading1    1   // 0x0001
#define stiHeading2    2   // 0x0002
#define stiHeading3    3   // 0x0003
#define stiHeading4    4   // 0x0004
#define stiHeading5    5   // 0x0005
#define stiHeading6    6   // 0x0006
#define stiHeading7    7   // 0x0007
#define stiHeading8    8   // 0x0008
#define stiHeading9    9   // 0x0009
#define stiHeadingFirst stiHeading1
#define stiHeadingLast  stiHeading9

#define stiIndex1      10  // 0x000A
#define stiIndex2      11  // 0x000B
#define stiIndex3      12  // 0x000C
#define stiIndex4      13  // 0x000D
#define stiIndex5      14  // 0x000E
#define stiIndex6      15  // 0x000F
#define stiIndex7      16  // 0x0010
#define stiIndex8      17  // 0x0011
#define stiIndex9      18  // 0x0012
#define stiIndexFirst  stiIndex1
#define stiIndexLast   stiIndex9

#define stiToc1        19  // 0x0013
#define stiToc2        20  // 0x0014
#define stiToc3        21  // 0x0015
#define stiToc4        22  // 0x0016
#define stiToc5        23  // 0x0017
#define stiToc6        24  // 0x0018
#define stiToc7        25  // 0x0019
#define stiToc8        26  // 0x001A
#define stiToc9        27  // 0x001B
#define stiTocFirst    stiToc1
#define stiTocLast     stiToc9
```

```
#define stiNormIndent  28  // 0x001C
#define stiFtnText     29  // 0x001D
#define stiAtnText     30  // 0x001E
#define stiHeader      31  // 0x001F
#define stiFooter      32  // 0x0020
#define stiIndexHeading 33 // 0x0021
#define stiCaption     34  // 0x0022
#define stiToCaption   35  // 0x0023
#define stiEnvAddr     36  // 0x0024
#define stiEnvRet      37  // 0x0025
#define stiFtnRef      38  // 0x0026  char style
#define stiAtnRef      39  // 0x0027  char style
#define stiLnn         40  // 0x0028  char style
#define stiPgn         41  // 0x0029  char style
#define stiEdnRef      42  // 0x002A  char style
#define stiEdnText     43  // 0x002B
#define stiToa         44  // 0x002C
#define stiMacro       45  // 0x002D
#define stiToaHeading  46  // 0x002E
#define stiList      47 // 0x002F
#define stiListBullet  48  // 0x0030
#define stiListNumber  49  // 0x0031
#define stiList2        50 // 0x0032
#define stiList3        51 // 0x0033
#define stiList4        52 // 0x0034
#define stiList5        53 // 0x0035
#define stiListBullet2 54  // 0x0036
#define stiListBullet3 55  // 0x0037
#define stiListBullet4 56  // 0x0038
#define stiListBullet5 57  // 0x0039
#define stiListNumber2 58  // 0x003A
#define stiListNumber3 59  // 0x003B
#define stiListNumber4 60  // 0x003C
#define stiListNumber5 61  // 0x003D
#define stiTitle     62 // 0x003E
#define stiClosing     63  // 0x003F
#define stiSignature   64  // 0x0040
#define stiNormalChar  65  // 0x0041  char style
#define stiBodyText    66  // 0x0042
#define stiBodyTextInd 67  // 0x0043
#define stiListCont    68  // 0x0044
#define stiListCont2   69  // 0x0045
#define stiListCont3   70  // 0x0046
#define stiListCont4   71  // 0x0047
#define stiListCont5   72  // 0x0048
#define stiMsgHeader   73  // 0x0049
#define stiSubtitle    74  // 0x004A
#define stiSalutation  75  // 0x004B
#define stiDate         76 // 0x004C
#define stiBodyText1I  77  // 0x004D
#define stiBodyText1I2 78  // 0x004E
#define stiNoteHeading 79  // 0x004F
#define stiBodyText2   80  // 0x0050
#define stiBodyText3   81  // 0x0051
```

```
#define stiBodyTextInd2 82 // 0x0052
#define stiBodyTextInd3 83 // 0x0053
#define stiBlockQuote  84  // 0x0054
#define stiHyperlink   85   // 0x0055  char style
#define stiHyperlinkFollowed 86 // 0x0056   char style
#define stiStrong      87   // 0x0057  char style
#define stiEmphasis    88   // 0x0058  char style
#define stiNavPane     89   // 0x0059  char style
#define stiPlainText   90   // 0x005A
#define stiAutoSig     91   // 0x005B
#define stiFormTop     92   // 0x005C
#define stiFormBottom  93   // 0x005D
#define stiHtmlNormal  94   // 0x005E
#define stiHtmlAcronym 95   // 0x005F  char style
#define stiHtmlAddress 96   // 0x0060
#define stiHtmlCite    97   // 0x0061  char style
#define stiHtmlCode    98   // 0x0062  char style
#define stiHtmlDfn     99   // 0x0063  char style
#define stiHtmlKbd    100   // 0x0064  char style
#define stiHtmlPre    101   // 0x0065
#define stiHtmlSamp   102   // 0x0066  char style
#define stiHtmlTt     103   // 0x0067  char style
#define stiHtmlVar    104   // 0x0068  char style
#define stiNormalTable 105 // 0x0069  table style
#define stiAtnSubject 106   // 0x0070
```

The following **Table** and **List** styles were added in Word 2002:

```
#define stiNormalList 107   // 0x0071  list style
#define stiOutlineList1 108 // 0x0072  list style (1 / a / i)
#define stiOutlineList2 109 // 0x0073  list style (1 / 1.1 / 1.1.1)
#define stiOutlineList3 110 // 0x0074  list style (Article / Section)
#define stiListStyleFirst stiNormalList   // First default list style
#define stiListStyleLast  stiOutlineList3 // Last default list style

#define stiTableSimple1      111
#define stiTableSimple2      112
#define stiTableSimple3      113
#define stiTableClassic1     114
#define stiTableClassic2     115
#define stiTableClassic3     116
#define stiTableClassic4     117
#define stiTableColorful1    118
#define stiTableColorful2    119
#define stiTableColorful3    120
#define stiTableColumns1     121
#define stiTableColumns2     122
#define stiTableColumns3     123
#define stiTableColumns4     124
#define stiTableColumns5     125
#define stiTableGrid1        126
#define stiTableGrid2        127
#define stiTableGrid3        128
#define stiTableGrid4        129
```

```
#define stiTableGrid5        130
#define stiTableGrid6        131
#define stiTableGrid7        132
#define stiTableGrid8        133
#define stiTableList1        134
#define stiTableList2        135
#define stiTableList3        136
#define stiTableList4        137
#define stiTableList5        138
#define stiTableList6        139
#define stiTableList7        140
#define stiTableList8        141
#define stiTable3DFx1        142
#define stiTable3DFx2        143
#define stiTable3DFx3        144
#define stiTableContemporary 145
#define stiTableElegant      146
#define stiTableProfessional 147
#define stiTableSubtle1      148
#define stiTableSubtle2      149
#define stiTableWeb1         150
#define stiTableWeb2         151
#define stiTableWeb3         152
#define stiTableFirst        stiTableSimple1
#define stiTableLast         stiTableWeb3
#define stiAcetate          153
#define stiTableGrid     154
#define stiTableTheme       155

#define stiMax        156  // number of defined sti's
#define stiChpMax      19  // number of defined char style sti's
#define stiPapMax      87  // number of defined para style sti's
#define stiTableMax     1  // number of defined table style sti's

#define stiUser        0x0ffe // user styles are distinguished by name#define
stiNil         0x0fff   // max for 12 bits
```

See below for the names of these styles.

std.stk: The type of each style is indicated by `std.stk`. The types currently in use are:

```
stkPara     1    A paragraph style
stkChar     2    A character style
stkTable    3    A table style
stkList     4    A list style
```

More style types may exist in the future, so styles of an unknown type should be discarded.

std.istdBase: The style that this style is based on. A style is always based on another style or the null style (`istdNil`). Following a "chain" of based-on styles will always end at the null style, because a based-on chain cannot have a loop in it. A style can have up to 11 "ancestors" in its based-on chain, including the null style. A style's definition is built up from the style that it is based on. See `std.cupx`, `std.grupx`, `std.grupe`.

std.istdNext: The style to apply after the current one. For a paragraph style, this is the style to apply when Enter is pressed at the end of a paragraph. For a character style, the next style is essentially ignored, but should be the same as the current style.

std.xstzName: The name of the style, including aliases. The name is stored as an `xstz` (preceded by a length byte, followed by a null-terminator.) A style name can contain multiple "aliases", separated by commas. Aliases are alternate names for the same style (e.g. a style named "a,b,c" has three aliases, and can be referred to by "a", "b", or "c", or any combination.) WinWord 2.x did not have aliases, but Word 5.x for the Macintosh did. If a style is a built-in style, the built-in style name is always stored first.

All names (and aliases) must be unique within a style sheet (e.g. styles "a,b" and "b,c" should not exist in the same style sheet, as "b" matches multiple style names.)

A style name (including all its aliases and comma separators) can be up to 253 characters long. So the `xstz` format of that name can be up to 255 characters. Style names are case sensitive.

The built-in style names (corresponding to each `sti` listed previously) are defined for each language version of Word. For English USA documents, the names are:

1 / 1.1 / 1.1.1	1 / a / i	Article / Section
Balloon Text	Block Text	Body Text
Body Text 2	Body Text 3	Body Text First Indent
Body Text First Indent 2	Body Text Indent	Body Text Indent 2
Body Text Indent 3	Caption	Closing
Comment Reference	Comment Subject	Comment Text
Date	Default Paragraph Font	Document Map
E-mail Signature	Emphasis	Endnote Reference
Endnote Text	Envelope Address	Envelope Return
FollowedHyperlink	Footer	Footnote Reference
Footnote Text	Header	Heading 1
Heading 2	Heading 3	Heading 4
Heading 5	Heading 6	Heading 7
Heading 8	Heading 9	HTML Acronym
HTML Address	HTML Cite	HTML Code
HTML Definition	HTML Keyboard	HTML Preformatted
HTML Sample	HTML Typewriter	HTML Variable
Hyperlink	Index 1	Index 2
Index 3	Index 4	Index 5
Index 6	Index 7	Index 8
Index 9	Index Heading	Line Number
List	List 2	List 3
List 4	List 5	List Bullet
List Bullet 2	List Bullet 3	List Bullet 4
List Bullet 5	List Continue	List Continue 2
List Continue 3	List Continue 4	List Continue 5
List Number	List Number 2	List Number 3
List Number 4	List Number 5	Macro Text
Message Header	No List	Normal
Normal (Web)	Normal Indent	Note Heading
Page Number	Plain Text	Salutation
Signature	Strong	Subtitle
Table 3D effects 1	Table 3D effects 2	Table 3D effects 3
Table Classic 1	Table Classic 2	Table Classic 3
Table Classic 4	Table Colorful 1	Table Colorful 2
Table Colorful 3	Table Columns 1	Table Columns 2
Table Columns 3	Table Columns 4	Table Columns 5
Table Contemporary	Table Elegant	Table Grid
Table Grid 1	Table Grid 2	Table Grid 3
Table Grid 4	Table Grid 5	Table Grid 6

```
Table Grid 7              Table Grid 8              Table List 1
Table List 2              Table List 3              Table List 4
Table List 5              Table List 6              Table List 7
Table List 8              Table Normal              Table of Authorities
Table of Figures          Table Professional        Table Simple 1
Table Simple 2            Table Simple 3            Table Subtle 1
Table Subtle 2            Table Theme               Table Web 1
Table Web 2               Table Web 3               Title
TOA Heading               TOC 1                     TOC 2
TOC 3                     TOC 4                     TOC 5
TOC 6                     TOC 7                     TOC 8
TOC 9
```

std.cupx: This is the number of UPXs in the std.grupx array. See below.

std.grupx: This is an array [More accurately a "group", because each of the elements (UPXs) in the array is variable-length] of variable-length UPXs, with std.cupx UPXs in the array. This array begins after the variable-length xstzName field, at the next even-byte offset within the STD. A UPX (Universal Property eXception) describes the difference in formatting of this style as compared to its based-on style. The UPX structure looks like this:

```
typedef union _UPX
{
   struct
      {
      uchar grpprl[cbMaxGrpprlStyleChpx];
      } chpx;
   struct
      {
      ushort istd;
      uchar grpprl[cbMaxGrpprlStylePapx];
      } papx;
   struct
      {
      uchar grpprl[cbMaxGrpprlForTaps * 8];  // enough for 8 full cnf's
      } tapx;
#ifdef STYLERM
   UPDRM rm;
#endif //STYLERM
   uchar  rgb[1];
} UPX;
```

Each UPX stored in a file is not a complete UPX, rather it is a UPX with all trailing zero bytes lopped off, and preceded by a ushort length field. So it is stored like:

Field	Size	Comment
cbUPX	2 bytes	Size of the following UPX structure
UPX	(cbUPX)	Nonzero prefix of a UPX structure

Each UPX begins on an even-byte offset within the STD, even if the length of the previous UPX (cbUPX) was odd.

The meaning of each UPX depends on the style type (std.stk). For a paragraph style, std.cupx=2. The first UPX is a paragraph UPX (UPX.papx) and the second UPX is a character

UPX (UPX.chpx). For a character style, std.cupx=1, and that UPX is a character UPX (UPX.chpx). Note that new UPXs may be added in the future, so std.cupx might be larger than expected. Any UPXs past those expected should be discarded. For a list style, std.cupx=1. The UPX is a paragraph UPX (UPX.papx). For a table style, std.cupx=3. The first UPX is a table UPX (UPX.tapx), the second UPX is a paragraph UPX (UPX.papx), and the third UPX is a character UPX (UPX.chpx). In addition, each style type can contain an additional UPX containing revision mark information, which is not documented.

The grpprl within each UPX contains the differences of this property type for this style from the UPE of that property type for the based on style. For example, if two paragraph styles, A and B, were identical except that B was bold where A was not, and B was based on A, B would have two UPXs, where the paragraph UPX would have an empty grpprl [Note that the UPX.papx contains both a grpprl and an istd. Even if the grpprl is empty, the istd is still needed.], and the character UPX would have a bold sprm in the grpprl. Thus B looks just like A (since B is based on A), with the exception that B is bold.

std.grupe: This is an array (group) of variable-length UPEs. **These are not stored in the file!** Rather, they are constructed using the std.istdBase and std.grupx fields. A UPE (Universal Property Expansion) describes the "end-result" of the property formatting, i.e. what the style looks like. The UPE structure is the non-zero prefix of a UPD structure. The UPD structure looks like this:

```
typedef union _UPD
   {
   PAP pap;
   CHP chp;
   TAPS taps;
   struct
      {
      ushort istd;
      uchar cbGrpprl;
      uchar grpprl[cbMaxGrpprlStyleChpx];
      } chpx;
   struct
      {
      ushort istd;
      uchar cbGrpprl;
      uchar grpprl[cbMaxGrpprlStylePapx];
      } papx;
#ifdef STYLERM
   UPDRM rm;
#endif //STYLERM
   } UPD;
```

The std.grupe and std.grupx arrays are similar: there is one UPE for each UPX, and internally they are stored similarly (a length ushort followed by a non-zero prefix). **Note**: UPEs are not stored in the file. The meaning of each UPE depends on the style type (std.sgc). For a paragraph style, the first UPE is a PAP (UPE.pap) and the second UPE is a CHP (UPE.chp). For a character style, the first UPE is a CHPX (UPE.chpx). List styles have one UPE, which is a PAPX (UPE.papx). For a table style the first UPE is a table UPE (UPE.taps), the second UPE is a paragraph UPE (UPE.pap), and the third UPE is a character UPE (UPE.chp). In addition, each style type can contain an additional UPE containing revision mark information, which is not documented.

The UPEs for a style are constructed by taking the UPEs from the based-on style, and applying the UPXs to them. If the UPEs for the based-on style haven't yet been constructed, that style's

UPE needs to be constructed first. Eventually by following the based-on chain, a style will be based on the null style (istdNil). The UPEs for the null style are predefined:

- The UPE.pap for the null style is all zeros, except fWidowControl which is 1, dyaLine which is 240, and fMultLinespace which is 1.
- The UPE.chp for the null style is all zeros, except istd which is 10 (istdNormalChar), hps which is 20, lid which is 0x0400, and ftc which is set to the STSHI.ftcStandardChpStsh.
- The UPE.chpx for the null style has an istd of zero, a cbGrpprl of zero (and an empty grpprl).

So, for a paragraph style, the first UPE is a UPE.pap. It can be constructed by starting with the first UPE from the based-on style (std.istdBase), and then applying the first UPX (UPX.papx) in std.grupx to that UPE. To apply a UPX.papx to a UPE.pap, set UPE.pap.istd equal to UPX.papx.istd, and then apply the UPX.papx.grpprl to UPE.pap. Similarly, the second UPE is a UPE.chp. It can be constructed by starting with the second UPE from the based-on style, and then applying the second UPX (UPX.chpx) in std.grupx to that UPE. To apply a UPX.chpx to a UPE.chp, apply the UPX.chpx.grpprl to UPE.chp. **Note**: a UPE.chp for a paragraph style should always have UPE.chp.istd==istdNormalChar.

For a character style, the first (and only) UPE (a UPE.chpx) can be constructed by starting with the first UPE from the based-on style (std.istdBase), and then applying the first UPX (UPX.chpx) in std.grupx to that UPE. To apply a UPX.chpx to a UPE.chpx, take the grpprl in UPE.chpx.grpprl (which has a length of UPE.chpx.cbGrpprl) and merge the grpprl in UPX.chpx.grpprl into it. Merging grpprls can be difficult, but for character styles it is easy because no prls in character style grpprls should interact with each other. Each prl from the source (the UPX.chpx.grpprl) should be inserted into the destination (the UPE.chpx.grpprl) so the sprm of each prl is in increasing order, and any prls with the same sprm are replaced by the prl in the source. UPE.chpx.cbGrpprl is then set to the length of resulting grpprl, and UPE.chpx.istd is set to the style's istd.

For a list style, the first (and only) UPE (a UPE.papx) can be constructed by starting with the first UPE from the based-on style (std.istdBase), and then applying the first UPX (UPX.papx) in std.grupx to that UPE. To apply a UPX.papx to a UPE.papx, take the grpprl in UPE.papx.grpprl (which has a length of UPE.papx.cbGrpprl) and merge the grpprl in UPX.papx.grpprl into it. Merging grpprls can be difficult. Each prl from the source (the UPX.papx.grpprl) should be inserted into the destination (the UPE.papx.grpprl) so the sprm of each prl is in increasing order, and any prls with the same sprm are replaced by the prl in the source. UPE.papx.cbGrpprl is then set to the length of resulting grpprl, and UPE.papx.istd is set to the style's istd.

So, for a table style, the first UPE is a UPE.taps. It can be constructed by starting with the first UPE from the based-on style (std.istdBase), and then applying the first UPX (UPX.tapx) in std.grupx to that UPE. To apply a UPX.tapx to a UPE.taps, set UPE.taps.istd equal to UPX.tapx.istd, and then apply the UPX.tapx.grpprl to UPE.taps. The second UPE is a UPE.pap. It can be constructed by starting with the first UPE from the based-on style (std.istdBase), and then applying the first UPX (UPX.papx) in std.grupx to that UPE. To apply a UPX.papx to a UPE.pap, set UPE.pap.istd equal to UPX.papx.istd, and then apply the UPX.papx.grpprl to UPE.pap. Similarly, the third UPE is a UPE.chp. It can be constructed by starting with the second UPE from the based-on style, and then applying the second UPX (UPX.chpx) in std.grupx to that UPE. To apply a UPX.chpx to a UPE.chp, apply the UPX.chpx.grpprl to UPE.chp. **Note**: a UPE.chp for a table style should always have UPE.chp.istd==istdNormalChar.

List Tables

Word 97 and later versions store paragraph numbering information very differently from Word 6.0. In Word 6.0, all information for a paragraph was stored in that paragraph's pap.anld. In Word 97 and later versions, the pap only contains two values: a short ilfo and a byte ilvl, which indicate which list the paragraph belongs to and which level of that list it is part of, respectively. The ilfo is actually an index into one of the document's list tables: the pllfo, and the paragraph gets most of its information about appearance from the list tables.

There are three list tables in a word document: the rglst, the hpllfo, and the hsttbListNames. They are described below in greater detail, and the precise formats of several of these structures (the LSTF, LVLF, LFO, and LFOLVL) are listed in the appendix.

LST Records and the rglst

The LST structure is where most of the list appearance data is stored. An LST consists of two main parts:

1. An LSTF, which is stored on disk and contains formatting properties which apply to the entire list, such as whether the list is simple or multilevel, the list's unique list index and template code, the istd's (see Stylesheet above) of the styles (if any) that each level in the list is linked to, and a number of Word 6.0 compatibility options.
2. An array of LVL structures, which describe the appearance of each individual level in the LST.

A LVL structure contains two parts:

1. An LVLF, which stores all static data such as the start-at value for the list level, the numbering type (arabic or roman), the alignment (left, right or centered) of the number, and several Word 6.0 compatibility options.
2. A set of pointers to variable length data:

   (a) a grpprlChpx, which sets character formatting to the paragraph number text,

   (b) a grpprlPapx, which sets paragraph formatting to the paragraph containing the number, such as indenting and tab information

   (c) the number text itself.

Word writes out the rglst as the plcflst by writing out a short integer containing the number of LST structures to be written; followed by an enumeration of the rglst, writing out each LSTF structure. It then enumerates through the rglst again, deciding, for each LST, whether it has one level (LSTF.fSimpleList) or nine levels (!LSTF.fSimpleList). It then writes the appropriate number of LVL structures as described below.

When Word writes out an LVL structure, it first writes out the LVLF, followed by the grpprlPapx (of LVLF.cbGrpprlPapx bytes in length), followed by the grpprlChpx (of length LVLF.cbGrpprlChpx), and an XCHAR string with the number text, preceded by an XCHAR containing the string's length.

List Names and the sttbListNames

The string table containing the List Names is by far the least significant of the three list tables. Most lists do not have names, and the names are only useful to users of Visual Basic for Applications (VBA). If this list has a name, however, it is in this table: the table is a parallel array with the rglst above, and will contain an empty string for any list which does not have a list name.

LFO Records and the pllfo

The LFO structure serves primarily as a level of indirection between the paragraph and the LST, but also can be used to override certain features of the list formats (LFO stands for List Format Override). An LFO consists of two main parts:

(1) the List ID of the list (LST record) to which this LFO belongs.

(2) an array of overrides to the formatting in that LST.

For the vast majority of LFOs, there are no overrides, but if there are any, they reside in an array of LFOLVL structures—one LFOLVL per level of the LST to be overridden. An LFOLVL contains a set of flags to indicate whether just the start-at value of the LST is overridden, or whether just the formatting is overridden, or both, as well as either a start-at value or a pointer to a LVL record, depending upon the values of the flags. **Note**: if the LFOLVL says the start-at value should be overridden, what that means is that the FIRST paragraph in the document with this LFO should have a number equal exactly to that start-at value, but any subsequent paragraphs should just follow the previous paragraph in the sequence. Also, if LFOLVL.fFormatting and LFOLVL.fStartAt are *both* true (rare) then LFOLVL.iStartAt should be ignored in favor of the iStartAt value from the corresponding LVL.

Word writes out the pllfo first by writing out a PL of LFO structures. It then enumerates through each LFO to figure out how many LFOLVLs each one has (LFO.clfolvl), and writes out, in order, each LFOLVL structure followed by its corresponding LVL structure (if LFOLVL.fFormatting is set).

Paragraph List Formatting

Given a paragraph and its corresponding PAP, the following process must be followed to find out the paragraph's list information:

1. Using the pap.ilfo, look up the LFO record in the pllfo with that (1-based) index.
2. Using the LFO, and the pap.ilvl, check to see if there are any overrides for this particular level. If so, and if the override pertains to both formatting and start-at value, use the LVL record from the correct LFOLVL in the LFO, and skip to step 5.
3. If the override does not pertain to either formatting or start-at value, look up the LST for this list. Using the LFO's List ID, search the rglst for the LST with that List ID.
4. Now, take from this LST any information (formatting or start-at value) still needed after consulting the LFO.
5. Once the correct LVL record is obtained, apply the lvl.grpprlPapx to the PAP. It may adjust the indents and tab settings for the paragraph.
6. Use the other information in the LVL, such as the start at, number text, and grpprlChpx, to determine the appearance of the actual paragraph number text.

SPRM Definitions

A sprm is an instruction to modify one or more properties within one of the property defining data structures (CHP, PAP, TAP, SEP, or PIC). A sprm is a two-byte opcode at offset 0 which identifies the operation to be performed. If necessary information for the operation can always be expressed with a fixed length parameter, the fixed length parameter is recorded immediately after the opcode beginning at offset 2. The length of a fixed length sprm is always 2 plus the size of the sprm's parameter. If the parameter for the sprm is variable length, the count of bytes of the following parameter is stored in the byte at offset 2, followed by the parameter at offset 3.

Three sprms -- sprmPChgTabs, sprmTDefTable, and sprmTDefTable10 -- can be longer than 256 bytes. The method for calculating the length of sprmPChgTabs is recorded below with the description of the sprm. For sprmTDefTable and sprmTDefTable10, the length of the parameter plus 1 is recorded in the two bytes beginning at offset 2.

For all other variable length sprms, the total length of the sprm is the count recorded at offset 2 plus three (2 for the sprm + 1 for the count byte). The parameter immediately follows the count.

The sprm value encodes information on the size of the operand, the type of sprm (PAP, CHP, etc), and whether the sprm requires special handling (in cases where a property value isn't simply replaced).

Sprm bits (0 = low)	Value	Details
0-8	ispmd	Unique identifier within sgc group
9	fSpec	sprm requires special handling
10-12	sgc	sprm group; type of sprm (PAP, CHP, etc)
13-15	spra	Size of sprm argument (see following table for values)

sgc value	Type of sprm
1	PAP
2	CHP
3	PIC
4	SEP
5	TAP

spra value	Operand size
0	1 byte (operand affects 1 bit)
1	1 byte
2	2 bytes
3	4 bytes
4	2 bytes
5	2 bytes
6	Variable length -- following byte is size of operand
7	3 bytes

When parsing a grpprl, you can use the sprm's spra value to determine how many bytes are used by that sprm; it also enables you to skip over sprms you don't handle.

Unless otherwise noted, when a sprm is applied to a property the sprm's parameter changes the old value of the property in question to the value stored in the sprm parameter.

Paragraph SPRMs

Name	sprm	Property modified	Parameter	Parameter size
sprmPIstd	0x4600	pap.istd	istd (style code)	short
sprmPIstdPermute	0xC601	pap.istd	permutation vector (see below)	variable length
sprmPIncLvl	0x2602	pap.istd, pap.lvl	difference between istd of base PAP and istd of PAP to be produced (see below)	byte
sprmPJc	0x2461	change pap.jc In Word 2000, justification is relative to text direction (left is left for left-to-right text and right for right-to-left text).	jc (justification)	spraByte
sprmPJc80	0x2403	change pap.jc (bi-directional Word 97 style)	jc (justification)	spraByte
sprmPFSideBySide	0x2404	pap.fSideBySide	0 or 1	byte
sprmPFKeep	0x2405	pap.fKeep	0 or 1	byte
sprmPFKeepFollow	0x2406	pap.fKeepFollow	0 or 1	byte
sprmPFPageBreakBefore	0x2407	pap.fPageBreakBefore	0 or 1	byte
sprmPBrcl	0x2408	pap.brcl	brcl	byte
sprmPBrcp	0x2409	pap.brcp	brcp	byte
sprmPIlvl	0x260A	pap.ilvl	ilvl	byte
sprmPIlfo	0x460B	pap.ilfo	ilfo (list index)	short
sprmPFNoLineNumb	0x240C	pap.fNoLnn	0 or 1	byte
sprmPChgTabsPapx	0xC60D	pap.itbdMac, pap.rgdxaTab, pap.rgtbd	complex (see below)	variable length
sprmPDxaLeft	0x845e	change pap.dxaLeft In Word 2000, dxaLeft is relative to text direction (see pap.dxaLeft definition).	dxa	Word (2 bytes)
sprmPDxaLeft80	0x840f	change pap.dxaLeft	dxa	word (2 bytes)
sprmPDxaLeft1	0x8460	change pap.dxaLeft1	dxa	word (2 bytes)

Name	sprm	Property modified	Parameter	Parameter size
sprmPDxaLeft180	0x8411	change pap.dxaLeft1 for Word 97	dxa	word (2 bytes)
sprmPDxaRight	0x845d	change pap.dxaRight In Word 2000, dxaRight is relative to text direction (see pap.dxaLeft definition).	dxa	word (2 bytes)
sprmPDxaRight80	0x840e	change pap.dxaRight (bi-directional Word 97 style)	dxa	word (2 bytes)
sprmPDxcLeft	0x4456	change pap.dxcLeft	dxa	word (2 bytes)
sprmPDxcLeft1	0x4457	change pap.dxcLeft1	dxa	word (2 bytes)
sprmPDxcRight	0x4455	change pap.dxcRight	dxa	word (2 bytes)
sprmPNest	0x465f	pap.dxaLeft	dxa (see below)	word (2 bytes)
sprmPNest80	0x4610	pap.dxaLeft	dxa (see below)	word (2 bytes)
sprmPDyaLine	0x6412	pap.lspd	an LSPD, a long word structure consisting of a short of dyaLine followed by a short of fMultLinespace (see below)	long
sprmPDyaBefore	0xA413	pap.dyaBefore	dya	word
sprmPDyaAfter	0xA414	pap.dyaAfter	dya	word
sprmPFDyaAfterAuto	0x245c	change pap.fDyaAfterAuto	1 or 0	byte
sprmPFDyaBeforeAuto	0x245b	change pap.fDyaBeforeAuto	1 or 0	byte
sprmPDylAfter	0x4459	change pap.dylAfter	short	word (2 bytes)
sprmPDylBefore	0x4458	change pap.dylBefore	short	word (2 bytes)
sprmPChgTabs	0xC615	pap.itbdMac, pap.rgdxaTab, pap.rgtbd	complex (see below)	variable length
sprmPFInTable	0x2416	pap.fInTable	0 or 1	byte
sprmPFTtp	0x2417	pap.fTtp	0 or 1	byte
sprmPDxaAbs	0x8418	pap.dxaAbs	dxa	word
sprmPDyaAbs	0x8419	pap.dyaAbs	dya	word
sprmPDxaWidth	0x841A	pap.dxaWidth	dxa	word
sprmPPc	0x261B	pap.pcHorz, pap.pcVert	complex (see below)	byte
sprmPBrcTop10	0x461C	pap.brcTop	BRC10	word

Name	sprm	Property modified	Parameter	Parameter size
sprmPBrcLeft10	0x461D	pap.brcLeft	BRC10	word
sprmPBrcBottom10	0x461E	pap.brcBottom	BRC10	word
sprmPBrcRight10	0x461F	pap.brcRight	BRC10	word
sprmPBrcBetween10	0x4620	pap.brcBetween	BRC10	word
sprmPBrcBar10	0x4621	pap.brcBar	BRC10	word
sprmPDxaFromText10	0x4622	pap.dxaFromText	dxa	word
sprmPWr	0x2423	pap.wr	wr (see description of PAP for definition)	byte
sprmPBrcBar	0xc653	change pap bar border	BRC	variable length
sprmPBrcBar70	0x4629	change pap bar border for Word 95 and earlier versions	BRC70	word (2 bytes)
sprmPBrcBar80	0x6629	change pap bar border for Word 97 and later versions	BRC80	long (4 bytes)
sprmPBrcBetween	0xc652	change pap between border	BRC	variable length
sprmPBrcBetween70	0x4428	change pap between border for Word 95 and earlier versions	BRC70	word (2 bytes)
sprmPBrcBetween80	0x6428	change pap between border for Word 97 and later versions	BRC80	long (4 bytes)
sprmPBrcBottom	0xc650	change pap bottom border	BRC	variable length
sprmPBrcBottom70	0x4426	change pap bottom border for Word 95 and earlier versions	BRC70	word (2 bytes)
sprmPBrcBottom80	0x6426	change pap bottom border for Word 97 and later versions	BRC80	long (4 bytes)
sprmPBrcLeft	0xc64f	change pap left border	BRC	variable length
sprmPBrcLeft70	0x4425	change pap left border for Word 95 and earlier versions	BRC70	word (2 bytes)
sprmPBrcLeft80	0x6425	change pap left border for Word 97 and later versions	BRC80	long (4 bytes)

Name	sprm	Property modified	Parameter	Parameter size
sprmPBrcRight	0xc651	change pap right border	BRC80	variable length
sprmPBrcRight70	0x4427	change pap right border for Word 95 or earlier versions	BRC	word (2 bytes)
sprmPBrcRight80	0x6427	change pap right border for Word 97 and later versions	BRC70	long (4 bytes)
sprmPBrcTop	0xc64e	change pap top border	BRC80	variable length
sprmPBrcTop70	0x4424	change pap top border for Word 95 or earlier versions	BRC70	word (2 bytes)
sprmPBrcTop80	0x6424	change pap top border for Word 97 and later versions	BRC80	long (4 bytes)
sprmPFNoAutoHyph	0x242A	pap.fNoAutoHyph	0 or 1	byte
sprmPWHeightAbs	0x442B	pap.wHeightAbs	w	word
sprmPDcs	0x442C	pap.dcs	DCS	short
sprmPShd80	0x442D	pap.shd for Word 97 and later versions	SHD	word
sprmPShd	0xc64d	change pap.shd	SHD	Variable length
sprmPDyaFromText	0x842E	pap.dyaFromText	dya	word
sprmPDxaFromText	0x842F	pap.dxaFromText	dxa	word
sprmPFLocked	0x2430	pap.fLocked	0 or 1	byte
sprmPFWidowControl	0x2431	pap.fWidowControl	0 or 1	byte
sprmPRuler	0xC632			variable length
sprmPFKinsoku	0x2433	pap.fKinsoku	0 or 1	byte
sprmPFWordWrap	0x2434	pap.fWordWrap	0 or 1	byte
sprmPFOverflowPunct	0x2435	pap.fOverflowPunct	0 or 1	byte
sprmPFTopLinePunct	0x2436	pap.fTopLinePunct	0 or 1	byte
sprmPFAutoSpaceDE	0x2437	pap.fAutoSpaceDE	0 or 1	byte
sprmPFAutoSpaceDN	0x2438	pap.fAutoSpaceDN	0 or 1	byte
sprmPWAlignFont	0x4439	pap.wAlignFont	iFa (see description of PAP for definition)	word

Name	sprm	Property modified	Parameter	Parameter size
sprmPFrameTextFlow	0x443A	pap.fVertical pap.fBackward pap.fRotateFont	complex (see description of PAP for definition)	word
sprmPISnapBaseLine	0x243B	Obsolete; not applicable in Word 97 and later versions.		byte
sprmPAnld80	0xC63E	pap.anld (Word 97)	ANLD80	variable length
sprmPAnldCv	0x6654	change pap.anld.anlv.cv	COLORREF sprmPAnld80 composed with sprmPAnldCv yields the ANLD	long (4 bytes)
sprmPPropRMark	0xC63F	pap.fPropRMark	complex (see below)	variable length
sprmPOutLvl	0x2640	pap.lvl	has no effect if pap.istd is < 1 or is > 9	byte
sprmPFBiDi	0x2441	Pap.fBiDi	1 or 0	byte
sprmPFNumRMIns	0x2443	pap.fNumRMIns	1 or 0	bit
sprmPNumRM	0xC645	pap.numrm	NUMRM	variable length
sprmPHugePapx	0x6645	see below	fc in the data stream to locate the huge grpprl (see below)	long
sprmPFUsePgsuSettings	0x2447	pap.fUsePgsuSettings	1 or 0	byte
sprmPFAdjustRight	0x2448	pap.fAdjustRight	1 or 0	byte
sprmPDtap	0x664a	Add the parameter to pap.itap and change pap.fInTableW97 (set it to 0 if pap.itap is 0 and to 1 otherwise).		long (4 bytes)
sprmPFInnerTableCell	0x244b	change Pap.fInnerTableCell	1 or 0	byte
sprmPFInnerTtp	0x244c	Word 97 compatibility indicates this end of paragraph mark is really an end of row marker for a nested table.	1 or 0	byte
sprmPFNoAllowOverlap	0x2462	change pap.fNoAllowOverlap	1 or 0	byte

Name	sprm	Property modified	Parameter	Parameter size
sprmPItap	0x6649	change pap.itap and pap.fInTableW97 (set it to 0 if pap.itap is 0 and to 1 otherwise)		long (4 bytes)
sprmPWall	0x2664	pap.fHasOldProps Used for paragraph property revision marking. The pap at the time fHasOldProps is set to 1, the is the old pap.	0 or 1	byte
sprmPIpgp	0x6465	pap.ipgp (HTML div ID for this paragraph)	div id	long
sprmPCnf	0xc666	pap.hplcnf (conditional paragraph properties)	cnfc (conditional format condition + grpprl of properties)	variable length
sprmPRsid	0x6467	Changes pap.rsid, a random number associated with paragraph formatting which improves the accuracy of Word's document merge feature.	rsid	long
sprmPIstdList	0x4468	pap.istdList (list style for this paragraph)	style	word
sprmPIstdListPermute	0xc669	pap.istdList (list style permute; see sprmPIstdPermute for permute definition)	see sprmPIstdPermute	variable length
sprmPDyaBeforeNotCp0	0xa46a	pap.dyaBefore (sets dyaBefore ONLY for para's not at the beginning of the document)	twips	uza
sprmPTableProps	0x646b	pap.tap (holds the native 2002 table properties; backward compatible props are stored after this FC value (and thus parsed by Word 2000)	FC (offset into data stream for properties)	long
sprmPTIstdInfo	0xc66c	pap.tap.yfti (information about the last table autofit conditional results)	YFTI	variable length

Name	sprm	Property modified	Parameter	Parameter size
sprmPFContextualSpacing	0x246d	pap.fContextualSpacing (collapses space between paragraphs of the same style)	0 or 1	byte
sprmPRpf	0x246e	pap.rpf (revision pane flags)	RPF	byte
sprmPPropRMark90	0xc66f	Changes pap.fPropRMark, pap.ibstPropRMark, pap.dttmPropRMark. Word 2000 sometimes crashes reading in Word2002 paragraph property revisions, so a new sprmPPropRMark was created for Word 2002 to use, and the old one was renamed sprmPPropRMark90 and is no longer generated.	Same as sprmPPropRMark.	variable length

Character SPRMs

Name	Sprm	Property modified	Parameter	Parameter size
sprmCFRMarkDel	0x0800	chp.fRMarkDel	1 or 0	bit
sprmCFRMark	0x0801	chp.fRMark	1 or 0	bit
sprmCFFldVanish	0x0802	chp.fFldVanish	1 or 0	bit
sprmCFSdtVanish	0x2A90	chp.fSdtVanish. Applies to xchSdtBegin ("<") and xchSdtEnd (">") characters to signify that they are "vanished" (hidden).	1 or 0	bit
sprmCPicLocation	0x6A03	chp.fcPic chp.fSpec	(see below)	variable length, length recorded is always 4
sprmCIbstRMark	0x4804	chp.ibstRMark	index into sttbRMark	short
sprmCDttmRMark	0x6805	chp.dttmRMark	DTTM	long
sprmCFData	0x0806	chp.fData	1 or 0	bit

Name	Sprm	Property modified	Parameter	Parameter size
sprmCIdslRMark	0x4807	chp.idslRMReason	an index to a table of strings defined in Word 6.0 executables	short
sprmCChs	0xEA08	chp.fChsDiff chp.chse	(see below)	3 bytes
sprmCSymbol	0x6A09	chp.fSpec, chp.xchSym, chp.ftcSym	(see below)	variable length, length recorded is always 4
sprmCFOle2	0x080A	chp.fOle2	1 or 0	bit
sprmCIdCharType	0x480B	obsolete; not applicable in Word 97 and later versions		
sprmCHighlight	0x2A0C	chp.fHighlight, chp.icoHighlight	ico (fHighlight is set to 1 if ico is not 0)	byte
sprmCObjLocation	0x680E	chp.fcObj in old documents	FC	long
sprmCObjpLocation	0x680e	change chp.fcObjp in Word 2000	FC	long
sprmCFFtcAsciSymb	0x2A10			
sprmCIstd	0x4A30	chp.istd	istd, see style sheet definition	short
sprmCIstdPermute	0xCA31	chp.istd	permutation vector (see below)	variable length
sprmCDefault	0x2A32	whole CHP (see below)	none	variable length
sprmCPlain	0x2A33	whole CHP (see below)	none	0
sprmCKcd	0x2A34			
sprmCFBold	0x0835	chp.fBold	0,1, 128, or 129 (see below)	byte
sprmCFItalic	0x0836	chp.fItalic	0,1, 128, or 129 (see below)	byte
sprmCFStrike	0x0837	chp.fStrike	0,1, 128, or 129 (see below)	byte
sprmCFOutline	0x0838	chp.fOutline	0,1, 128, or 129 (see below)	byte
sprmCFShadow	0x0839	chp.fShadow	0,1, 128, or 129 (see below)	byte
sprmCFSmallCaps	0x083A	chp.fSmallCaps	0,1, 128, or 129 (see below)	byte

Name	Sprm	Property modified	Parameter	Parameter size
sprmCFCaps	0x083B	chp.fCaps	0,1, 128, or 129 (see below)	byte
sprmCFVanish	0x083C	chp.fVanish	0,1, 128, or 129 (see below)	byte
sprmCFtcDefault	0x4A3D		ftc, only used internally, never stored in file	word
sprmCKul	0x2A3E	chp.kul	kul	byte
sprmCSizePos	0xEA3F	chp.hps, chp.hpsPos	(see below)	3 bytes
sprmCDxaSpace	0x8840	chp.dxaSpace	dxa	word
sprmCLid	0x4A41		only used internally never stored	word
sprmCIco	0x2A42	chp.ico for Word 97 and later versions	ico	byte
sprmCHps	0x4A43	chp.hps	hps	byte
sprmCHpsInc	0x2A44	chp.hps	(see below)	byte
sprmCHpsPos	0x4845	chp.hpsPos	hps	byte
sprmCHpsPosAdj	0x2A46	chp.hpsPos	hps (see below)	byte
sprmCMajority	0xCA47	chp.fBold, chp.fItalic, chp.fSmallCaps, chp.fVanish, chp.fStrike, chp.fCaps, chp.rgftc, chp.hps, chp.hpsPos, chp.kul, chp.dxaSpace, chp.ico, chp.rglid chp.fOutline chp.fShadow chp.ftc chp.cv	complex (see below)	variable length, length byte plus size of following grpprl
sprmCIss	0x2A48	chp.iss	iss	byte
sprmCHpsNew50	0xCA49	chp.hps	hps	variable width, length always recorded as 2
sprmCHpsInc1	0xCA4A	chp.hps	complex (see below)	variable width, length always recorded as 2
sprmCHpsKern	0x484B	chp.hpsKern	hps	short

Name	Sprm	Property modified	Parameter	Parameter size
sprmCMajority50	0xCA4C	chp.fBold, chp.fItalic, chp.fSmallCaps, chp.fVanish, chp.fStrike, chp.fCaps, chp.ftc, chp.hps, chp.hpsPos, chp.kul, chp.dxaSpace, chp.ico	complex (see below)	variable length
sprmCHpsMul	0x4A4D	chp.hps	percentage to grow hps	short
sprmCHresi	0x484e	change chp.hresi (chp.hres, chp.chHres)	HRESI	word (2 bytes)
sprmCRgFtc0	0x4A4F	chp.rgftc[0]	ftc for ASCII text (see below)	short
sprmCRgFtc1	0x4A50	chp.rgftc[1]	ftc for East Asian text (see below)	short
sprmCRgFtc2	0x4A51	chp.rgftc[2]	ftc for non-East Asian text (see below)	short
sprmCCharScale	0x4852	chp.pctCharWidth		word (2 bytes)
sprmCFDStrike	0x2A53	chp.fDStrike		byte
sprmCFImprint	0x0854	chp.fImprint	1 or 0	bit
sprmCFSpec	0x0855	chp.fSpec	1 or 0	bit
sprmCFObj	0x0856	chp.fObj	1 or 0	bit
sprmCPropRMark	0xCA57	chp.fPropRMark, chp.ibstPropRMark, chp.dttmPropRMark	Complex (see below)	variable length always recorded as 7 bytes
sprmCFEmboss	0x0858	chp.fEmboss	1 or 0	bit
sprmCSfxText	0x2859	chp.sfxtText	text animation	byte
sprmCFBiDi	0x085A	Change chp.fBidi	1 or 0	bit
sprmCFDiacColor	0x085B		1 or 0	bit
sprmCFBoldBi	0x085C	Change chp.fBoldBi	1 or 0	bit
sprmCFItalicBi	0x085D	Change chp.fItalicBi	1 or 0	bit
sprmCFtcBi	0x4A5E	Change chp.ftcBi	ftc	word
sprmCLidBi	0x485F	Change chp.rglid[2] (chp.lidBi)	LID	word
sprmCIcoBi	0x4A60	Change chp.IcoBi		word
sprmCHpsBi	0x4A61	Change chp.HpsBi	Font size	word

Name	Sprm	Property modified	Parameter	Parameter size
sprmCDispFldRMark	0xCA62	chp.fDispFldRMark, chp.ibstDispFldRMark, chp.dttmDispFldRMark	Complex (see below)	variable length always recorded as 39 bytes
sprmCIbstRMarkDel	0x4863	chp.ibstRMarkDel	index into sttbRMark	short
sprmCDttmRMarkDel	0x6864	chp.dttmRMarkDel	DTTM	long
SprmCBrc80	0x6865	chp.brc for Word 97	BRC80	long
sprmCBrc	0xca72	change chp.brc	BRC	variable length
sprmCShd80	0x4866	chp.shd	SHD80	short
sprmCShd	0xca71	change chp.shd	SHD	variable length
sprmCIdslRMarkDel	0x4867	chp.idslRMReasonDel	an index to a table of strings defined in Word 6.0 executables	short
sprmCFUsePgsuSettings	0x0868	chp.fUsePgsuSettings	1 or 0	bit
sprmCCpg	0x486B			word
sprmCRgLid0_80	0x486D	chp.rglid[0] for Word 97	LID	word
sprmCRgLid0	0x4873	change chp.rglid[0]	LID	word
sprmCRgLid1_80	0x486E	chp.rglid[1] for Word 97	LID	word
sprmCRgLid1	0x4874	change chp.rglid[1]	LID	word
sprmCIdctHint	0x286F	chp.idctHint	IDCT: (see below)	byte
sprmCCv	0x6870	change chp.cv	COLORREF	long (4 bytes)
sprmCCvPermute	0xca7c	permute colors	complex (see below)	variable length
sprmCCvUl	0x6877	change chp.cvUl	COLORREF	long (4 bytes)
sprmCFBoldPresent	0x287d	change chp.fBoldPresent	1 or 0	byte
sprmCFELayout	0xca78	East Asian Warichu, Tatenakayoko and Kumimoji	complex	variable length
sprmCFItalicPresent	0x287e	change chp.fItalicPresent	1 or 0	byte
sprmCFitText	0xca76	change chp.dxaFitText & chp.lFitTextID	complex	variable length
sprmCFLangApplied	0x2a7a	change chp.fLangApplied (Abandoned)	1 or 0	byte
sprmCFNoProof	0x875	change chp.fNoProof	1 or 0	bit
sprmCFWebHidden	0x811	change chp.fWebHidden	1 or 0	bit

Name	Sprm	Property modified	Parameter	Parameter size
sprmCHsp	0x6a12	change `chp.fcPic` and set `fSpec`	complex	long (4 bytes)
sprmCLbcCRJ	0x2879	change `chp.lbrCRJ`	unsigned char	byte
sprmCNewIbstRM	0xca13	change `chp.ibstRMark` if original is not a threading author	IBST	variable length
sprmCTransNoProof0	0x287f	change `chp.bTransNoProof0`	REMOVE?	spraByte
sprmCTransNoProof1	0x2880	change `chp.bTransNoProof1`	REMOVE?	spraByte
sprmCFRMMove	0x2814	Not used	Not used	byte
sprmCRsidProp	0x6815	Changes `chp.rsidProp`, a random number associated with character formatting which improves the accuracy of Word's document merge feature.	rsid	long
sprmCRsidText	0x6816	Changes `chp.rsidText`, a random number associated with the insertion of text which improves the accuracy of Word's document merging.	rsid	long
sprmCRsidRMDel	0x6817	Changes `chp.rsidRMDel`, a random number associated with the tracked deletion of text which improves the accuracy of Word's document merging.	rsid	long
sprmCFSpecVanish	0x0818	`chp.fSpecVanish`	0 or 1	bit
sprmCFComplexScripts	0x0882	`chp.fComplexScripts`	1 or 0	bit
sprmCWall	0x2a83	`chp.fHasOldProps` Used for character property revision marking. The `chp` at the time `fHasOldProps` is set to 1, is the old `chp`.	0 or 1	byte
sprmCPbi	0xca84	`chp.pbi` (picture bullet information)		variable length

Name	Sprm	Property modified	Parameter	Parameter size
sprmCCnf	0xca85	chp.hplcnf (conditional character formatting for table styles. No language properties are stored here)		variable length
sprmCNeedFontFixup	0x2a86	chp.ffm	For internal use only, should never be seen in binary document	byte
sprmCPbiIBullet	0x6887	chp.pbi (picture bullet information)		long
sprmCPbiGrf	0x4888	chp.pbi (picture bullet information)		word
sprmCPropRMark	0xca89	chp.fPropRMark, chp.ibstPropRMark, chp.dttmPropRMark	Same as sprmPPropRMark	variable length

Picture SPRMs

Name	Sprm	Property modified	Parameter	Parameter size
sprmPicBrcl	0x2E00	pic.brcl	brcl (see PIC structure definition)	Byte
sprmPicScale	0xCE01	pic.mx, pic.my, pic.dxaCropleft, pic.dyaCropTop, pic.dxaCropRight, pic.dyaCropBottom	complex (see below)	length byte plus 12 bytes
sprmPicBrcTop80	0x6C02	pic.brcTop for Word 97	BRC80	long (4 bytes)
sprmPicBrcBottom	0xce0a	change pic bottom border	BRC	variable length
sprmPicBrcBottom70	0x4c04	change pic bottom border for Word 95 and earlier versions	BRC70	word (2 bytes)
sprmPicBrcLeft80	0x6C03	pic.brcLeft for Word 97	BRC80	long (4 bytes)
sprmPicBrcLeft	0xce09	change pic left border	BRC	variable length
sprmPicBrcLeft70	0x4c03	change pic left border for Word 95 and earlier versions	BRC70	word (2 bytes)
sprmPicBrcBottom80	0x6C04	pic.brcBottom for Word	BRC80	long (4 bytes)
sprmPicBrcRight	0xce0b	change pic right border	BRC	variable length
sprmPicBrcRight70	0x4c05	change pic right border for Word 95 and earlier versions	BRC70	word (2 bytes)

Name	Sprm	Property modified	Parameter	Parameter size
sprmPicBrcRight80	0x6C05	pic.brcRight for Word 97	BRC80	long (4 bytes)
sprmPicBrcTop	0xce08	change pic top border	BRC	variable length
sprmPicBrcTop70	0x4c02	change pic top border for Word 95 and earlier versions	BRC70	word (2 bytes)
sprmPicSpare4	0xce06			
sprmCFOle2WasHere	0xce07			

Section SPRMs

Name	Sprm	Property modified	Parameter	Parameter size
sprmScnsPgn	0x3000	sep.cnsPgn	cns	Byte
sprmSiHeadingPgn	0x3001	sep.iHeadingPgn	heading number level	Byte
sprmSOlstAnm	0xD202	sep.olstAnm	OLST	variable length
sprmSOlstAnm80	0xd202	sep.olstAnm for Word 97	OLST	variable length
sprmSOlstCv	0xd238	change Sep.olst.rganlv[9].cv	COLORREF[9], one for each ANLV in the OLST	variable length
sprmSDxaColWidth	0xF203	sep.rgdxaColWidthSpacing	complex (see below)	3 bytes
sprmSDxaColSpacing	0xF204	sep.rgdxaColWidthSpacing	complex (see below)	3 bytes
sprmSFEvenlySpaced	0x3005	sep.fEvenlySpaced	1 or 0	byte
sprmSFProtected	0x3006	sep.fUnlocked	1 or 0	byte
sprmSDmBinFirst	0x5007	sep.dmBinFirst		word
sprmSDmBinOther	0x5008	sep.dmBinOther		word
sprmSBkc	0x3009	sep.bkc	bkc	byte
sprmSFTitlePage	0x300A	sep.fTitlePage	0 or 1	byte
sprmSCcolumns	0x500B	sep.ccolM1	# of cols - 1	word
sprmSDxaColumns	0x900C	sep.dxaColumns	dxa	word
sprmSFAutoPgn	0x300D	sep.fAutoPgn	obsolete	byte
sprmSNfcPgn	0x300E	sep.nfcPgn	nfc	byte
sprmSDyaPgn	0xB00F	sep.dyaPgn	dya	short
sprmSDxaPgn	0xB010	sep.dxaPgn	dya	short
sprmSFPgnRestart	0x3011	sep.fPgnRestart	0 or 1	byte

Name	Sprm	Property modified	Parameter	Parameter size
sprmSFEndnote	0x3012	sep.fEndnote	0 or 1	byte
sprmSLnc	0x3013	sep.lnc	lnc	byte
sprmSGprfIhdt	0x3014	sep.grpfIhdt	grpfihdt (see Headers and Footers topic)	byte
sprmSNLnnMod	0x5015	sep.nLnnMod	non-neg int.	word
sprmSDxaLnn	0x9016	sep.dxaLnn	dxa	word
sprmSDyaHdrTop	0xB017	sep.dyaHdrTop	dya	word
sprmSDyaHdrBottom	0xB018	sep.dyaHdrBottom	dya	word
sprmSLBetween	0x3019	sep.fLBetween	0 or 1	byte
sprmSVjc	0x301A	sep.vjc	vjc	byte
sprmSLnnMin	0x501B	sep.lnnMin	lnn	word
sprmSPgnStart	0x501C	sep.pgnStart	pgn	word
sprmSBOrientation	0x301D	sep.dmOrientPage	dm	byte
sprmSXaPage	0xB01F	sep.xaPage	xa	word
sprmSYaPage	0xB020	sep.yaPage	ya	word
sprmSDxaLeft	0xB021	sep.dxaLeft	dxa	word
sprmSDxaRight	0xB022	sep.dxaRight	dxa	word
sprmSDyaTop	0x9023	sep.dyaTop	dya	word
sprmSDyaBottom	0x9024	sep.dyaBottom	dya	word
sprmSDzaGutter	0xB025	sep.dzaGutter	dza	word
sprmSDmPaperReq	0x5026	sep.dmPaperReq	dm	word
sprmSPropRMark	0xD227	sep.fPropRMark, sep.ibstPropRMark, sep.dttmPropRMark	complex (see below)	variable length always recorded as 7 bytes
sprmSFBiDi	0x3228	Change sep.fbidi	0 or 1	byte
sprmSFFacingCol	0x3229	Change sep.ffacingcol	0 or 1	byte
sprmSFRTLGutter	0x322A	Change sep.fRTLgutter	0 or 1	byte
sprmSBrcTop80	0x702B	sep.brcTop for Word 97	BRC	long (4 bytes)
sprmSBrcTop	0xd234	change sep.brcTop	BRC	variable length
sprmSBrcLeft80	0x702C	sep.brcLeft for Word 97	BRC	long (4 bytes)
sprmSBrcLeft	0xd235	change sep.brcLeft	BRC	variable length
sprmSBrcBottom80	0x702d	change sep.brcBottom for Word 97	BRC80	long (4 bytes)
sprmSBrcBottom	0xd236	change sep.brcBottom	BRC	variable length

Name	Sprm	Property modified	Parameter	Parameter size
sprmSBrcRight80	0x702e	change `sep.brcRight` for Word 97	`BRC80`	long (4 bytes)
sprmSBrcRight	0xd237	change `sep.brcRight`	`BRC`	long (4 bytes)
sprmSPgbProp	0x522F	`sep.pgbProp`		word
sprmSDxtCharSpace	0x7030	`sep.dxtCharSpace`	dxt	long
sprmSDyaLinePitch	0x9031	`sep.dyaLinePitch`	dya	long
sprmSClm	0x5032	`sep.clm`		word (2 bytes)
sprmSTextFlow	0x5033	`sep.wTextFlow`	complex (see below)	Short
sprmSWall	0x3239	`sep.fHasOldProps` Used for section property revision marking. The `sep` at the time `fHasOldProps` is set to 1, the is the old `sep`.	0 or 1	byte
sprmSRsid	0x703a	Change `sep.rsid`, a random number associated with section formatting which improves the accuracy of Word's document merging.	rsid	long
sprmSFpc	0x303b	`sep.fpc` (footnote position code)	fpc	byte
sprmSRncFtn	0x303c	`sep.fncFtn` (restart numbering code for footnotes)	rnc	byte
sprmSEpc	0x303d	`sep.epc` (endnode positioning code)	epc	byte
sprmSRncEdn	0x303e	`sep.rncEdn` (restart numbering code for endnotes)	rnc	byte
sprmSNFtn	0x503f	`sep.nFtn` (starting footnote number)	number	word
sprmSNfcFtnRef	0x5040	`sep.nfcFtnRef` (number format for footnote references)	nfc	word
sprmSNEdn	0x5041	`sep.nEdn` (starting endnote number)	number	word
sprmSNfcEdnRef	0x5042	`sep.nfcEdnRef` (number format for endnote references)	nfc	word
sprmSPropRMark	0xd243	`sep.fPropRMark`, `sep.ibstPropRMark`, `sep.dttmPropRMark`	Same as `sprmPPropRMark`	variable length

Table sprms

Name	Sprm	Property modified	Parameter	Parameter size
sprmTDefTable	0xD608	tap.rgtc for Word 97	complex (see below)	
sprmTDefTable10	0xD606	tap.rgdxaCenter, tap.rgtc	complex (see below)	variable length
sprmTDefTableShd	0xD609	change tap.rgshd for 97	complex (see below)	
sprmTDefTableShd	0xd612	change tap.rgtc[].shd cols 0 - 21	complex (see below)	variable length
sprmTDefTableShd2nd	0xd616	change tap.rgtc[].shd cols 22 - 43	complex (see below)	variable length
sprmTDefTableShd3rd	0xd60c	change tap.rgtc[].shd cols 44 - 63	complex (see below)	variable length
sprmTDelete	0x5622	tap.rgdxaCenter, tap.rgtc	complex (see below)	Word
sprmTDiagLine	0xd630	set BRC values for diagonal line in table cell (East Asian)	complex (see below)	variable length
sprmTDiagLine80	0xd62a	set BRC80 values for diagonal line in table cell (East Asian)	complex (see below)	variable length
sprmTDxaCol	0x7623	tap.rgdxaCenter	complex (see below)	4 bytes
sprmTDxaGapHalf	0x9602	tap.dxaGapHalf, tap.rgdxaCenter (see below)	dxa	word
sprmTDxaLeft	0x9601	tap.rgdxaCenter (see below)	dxa	word
sprmTDyaRowHeight	0x9407	tap.dyaRowHeight	dya	word
sprmTFBiDi80	0x560b	Tap.fBidi	0 or 1	word (2 bytes)
sprmTFCantSplit	0x3403	tap.fCantSplit	1 or 0	byte
sprmTHTMLProps	0x740C			
sprmTInsert	0x7621	tap.rgdxaCenter, tap.rgtc	complex (see below)	4 bytes
sprmTJc	0x5400	tap.jc	jc	word (low order byte is significant)
sprmTMerge	0x5624	tap.fFirstMerged, tap.fMerged	complex (see below)	word
sprmTSetBrc80	0xD620	tap.rgtc[].rgbrc for Word 97	complex (see below)	5 bytes
sprmTSetBrc10	0xD626	tap.rgtc[].rgbrc	complex (see below)	5 bytes
sprmTSetBrc	0xd62f	tap.rgtc[].rgbrc	complex (see below)	variable length
sprmTSetShd80	0x7627	tap.rgshd for Word 97	complex (see below)	4 bytes
sprmTSetShdOdd80	0x7628	tap.rgshd for Word 97	complex (see below)	4 bytes

Name	Sprm	Property modified	Parameter	Parameter size
sprmTSetShd	0xd62d	change `tap.rgtc[].shd`	complex (see below)	variable length
sprmTSetShdOdd	0xd62e	change `tap.rgtc[].shd`	complex (see below)	variable length
sprmTSetShdTable	0xd660	change `tap.shdTable`	SHD	variable length
sprmTSplit	0x5625	`tap.fFirstMerged`, `tap.fMerged`	complex (see below)	word
sprmTTableBorders	0xd613	change tap.rgbrcTable	`BRC[6]` (see below)	variable length
sprmTTableBorders80	0xd605	change `tap.rgbrcTable` for Word 97	`BRC80[6]` (see below)	variable length
sprmTTableHeader	0x3404	`tap.fTableHeader`	1 or 0	byte

sprmTTextFlow	0x7629	`tap.rgtc[].fVertical`
		`tap.rgtc[].fBackward`
		`tap.rgtc[].fRotateFont`
			`0 or 1`
			`0 or 1`
			`0 or 1`
				word

sprmTTlp	0x740A	`tap.tlp`	TLP	4 bytes
sprmTVertAlign	0xD62C	`tap.rgtc[].vertAlign`	complex (see below)	variable length always recorded as 3 byte
sprmTVertMerge	0xD62B	`tap.rgtc[].vertMerge`	complex (see below)	variable length always recorded as 2 bytes
sprmTFCellNoWrap	0xd639	change `tc.fNoWrap`	1 or 0	variable length
sprmTFitText	0xf636	change `FitText` setting in `TC`s	1 or 0	3 bytes
sprmTFKeepFollow	0x3619	change `tap.fKeepFollow`	1 or 0	byte
sprmTFNeverBeenAutofit	0x3663	change `tap.fNeverBeenAutofit`	1 or 0	byte
sprmTFNoAllowOverlap	0x3465	change `tap.fNoAllowOverlap`	1 or 0	byte
sprmTPc	0x360d	change positioning code	complex (see below)	byte
sprmTBrcBottomCv	0xd61c	set `tap.rgtc[].rgbrc[ibrcBottom].cv` for cols 0 - 63	complex (see below)	variable length
sprmTBrcLeftCv	0xd61b	set `tap.rgtc[].rgbrc[ibrcLeft].cv` for cols 0 - 63	complex (see below)	variable length
sprmTBrcRightCv	0xd61d	set `tap.rgtc[].rgbrc[ibrcRight].cv` for cols 0 - 63	complex (see below)	variable length
sprmTBrcTopCv	0xd61a	`tap.rgtc[].set rgbrc[ibrcTop].cv` for cols 0 - 63	complex (see below)	variable length

Name	Sprm	Property modified	Parameter	Parameter size
sprmTCellBrcType	0xd662	change `tap.rgtc[].brcLeft.brcType`, `tap.rgtc[].brcBottom.brcType`, `tap.rgtc[].brcRight.brcType`, `tap.rgtc[].brcTop.brcType`	complex (see below)	variable length
sprmTCellPadding	0xd632	change `tc.mpibrcwCellSpacing` and `tc.mpibrcftsCellSpacing`	complex (see below)	variable length
sprmTCellPaddingDefault	0xd634	change `tap.mpibrcwCellSpacingDefault` and `tap.mpibrcftsCellSpacingDefault`	complex (see below)	variable length
sprmTCellPaddingOuter	0xd638	change `tap.mpibrcwCellSpacingOuter` and `tap.mpibrcftsCellSpacingOuter`	complex (see below)	variable length
sprmTCellSpacing	0xd631	change `tc.mpibrcwCellSpacing` and `tc.mpibrcftsCellSpacing`	complex (see below)	variable length
sprmTCellSpacingDefault	0xd633	change `tap.mpibrcwCellSpacingDefault` and `tap.mpibrcftsCellSpacingDefault`	complex (see below)	variable length
sprmTCellSpacingOuter	0xd637	change `tap.mpibrcwCellSpacingOuter` and `tap.mpibrcftsCellSpacingOuter`	complex (see below)	variable length
sprmTCellWidth	0xd635	change width `tc.wWidth` and `tc.ftsWidth`	complex (see below)	variable length
sprmTDxaAbs	0x940e	change `tap.dxaAbs`	dxa	word (2 bytes)
sprmTDxaFromText	0x9410	change `tap.dxaFromText`	dxa	word (2 bytes)
sprmTDxaFromTextRight	0x941e	change `tap.dxaFromTextRight`	dxa	word (2 bytes)
sprmTDyaAbs	0x940f	change `tap.dyaAbs`	dxa	word (2 bytes)

Name	Sprm	Property modified	Parameter	Parameter size
sprmTDyaFromText	0x9411	change tap.dxaFromText	dya	word (2 bytes)
sprmTDyaFromTextBottom	0x941f	change tap.dxaFromTextBottom	dya	word (2 bytes)
sprmTFAutofit	0x3615	change fAutofit in TAP	1 or 0	byte
sprmTTableWidth	0xf614	change tap.ftsWidth and tap.wWidth	complex (see below)	3 bytes
sprmTWidthAfter	0xf618	change tap.ftsWidthAfter and tap.wWidthAfter	complex (see below)	3 bytes
sprmTWidthBefore	0xf617	change tap.ftsWidthBefore and tap.wWidthBefore	complex (see below)	3 bytes
sprmTWidthIndent	0xf661	change tap.ftsWidthIndent and tap.wWidthIndent in TAP	complex (see below)	3 bytes
sprmTIstd	0x563a	tap.istd (table style; no language props here)	long	word
sprmTSetShdRaw	0xd63b	tap.rgtc[].shd (user applied cell shading)	[itcFirst[1], itcLim[1], brck[1], SHD]	variable length
sprmTSetShdOddRaw	0xd63c	tap.rgtc[].shd (user applied odd cell shading)	see above	variable length
sprmTIstdPermute	0xd63d	tap.istd (table style permute, see sprmPIstdPermute for info on permutes)	style ID	variable length
sprmTCellPaddingStyle	0xd63e	tap.tcDefault.mpibrcwCellPadding/mpibrcftsCellPadding (cell padding for table style definitions)	see sprmTCellPading	variable length
sprmTFCantSplit90	0x3466	tap.fCantSplit90 Word 2002 allows a table row with vert merge cells to be broken across pages. That would sometimes cause Word 97 to crash, so a new sprmTFCantSplit was created and the old one renamed sprmTFCantSplit90 is used to tell Word 97 and Word 2000 not to break such a table row.	0 or 1	byte
sprmTPropRMark	0xd667	tap.fPropRMark, tap.ibstPropRMark, tap.dttmPropRMark.	Same as sprmPPropRMark	variable length

Name	Sprm	Property modified	Parameter	Parameter size
sprmTWall	0x3668	`tap.fHasOldProps` Used for table property revision marking. The `tap` at the time `fHasOldProps` is set to 1, the is the old `tap`.	0 or 1	byte
sprmTIpgp	0x7469	`tap.ipgp` (DIV ID for HTML Div Borders & Margins)	div ID	long
sprmTCnf	0xd66a	`tap.hplcnf` (conditional table formatting)	see sprmPCnf	variable length
sprmTSetShdTableDef	0xd66b	`tap.shdTableDef` (calculated default row shading)	SHD	variable length
sprmTDiagLine2nd	0xd66c	`tap.rgtc[14..29].tcd` (diagonal table borders)	[`BRC`, `BRC`] * count of cells	variable length
sprmTDiagLine3rd	0xd66d	`tap.rgtc[30..45].tcd` (diagonal table borders)	[`BRC`, `BRC`] * count of cells	variable length
sprmTDiagLine4th	0xd66e	`tap.rgtc[46..60].tcd` (diagonal table borders)	[`BRC`, `BRC`] * count of cells	variable length
sprmTDiagLine5th	0xd66f	`tap.rgtc[60..63].tcd` (diagonal table borders)	[`BRC`, `BRC`] * count of cells	variable length
sprmTDefTableShdRaw	0xd670	`tap.rgtc[0..21].shdRaw` (user defined default row shading)	array of SHD	variable length
sprmTDefTableShdRaw2nd	0xd671	`tap.rgtc[22..43].shdRaw` (user defined default row shading)	array of `SHD`	variable length
sprmTDefTableShdRaw3rd	0xd672	`tap.rgtc[44..63].shdRaw` (user defined default row shading)	array of `SHD`	variable length
sprmTSetShdRowFirst	0xd673	deprecated; not used		variable length
sprmTSetShdRowLast	0xd674	deprecated; not used		variable length
sprmTSetShdColFirst	0xd675	deprecated; not used		variable length
sprmTSetShdColLast	0xd676	deprecated; not used		variable length
sprmTSetShdBand1	0xd677	deprecated; not used		variable length
sprmTSetShdBand2	0xd678	deprecated; not used		variable length
sprmTRsid	0x7479	Change `tap.rsid`, a random number associated with table formatting which improves the accuracy of Word's document merging.	rsid	long
sprmTCellWidthStyle	0xf47a	deprecated; not used		tribyte

Name	Sprm	Property modified	Parameter	Parameter size
sprmTCellPaddingStyleBad	0xd67b	deprecated; not used		variable length
sprmTCellVertAlignStyle	0x347c	tap.tcDefault.vertAlign	(0,1,2,3) => (vaTop, vaCenter, vaBottom, vaJustify)	byte
sprmTCellNoWrapStyle	0x347d	tap.tcDefault.fNoWrap (don't wrap words in this cell)	0 or 1	byte
sprmTCellFitTextStyle	0x347e	deprecated; not used		byte
sprmTCellBrcTopStyle	0xd47f	tap.tcDefault.brcTop (border definition)	BRC	variable length
sprmTCellBrcBottomStyle	0xd680	tap.tcDefault.brcBottom (border definition)	BRC	variable length
sprmTCellBrcLeftStyle	0xd681	tap.tcDefault.brcLeft (border definition)	BRC	variable length
sprmTCellBrcRightStyle	0xd682	tap.tcDefault.brcRight (border definition)	BRC	variable length
sprmTCellBrcInsideHStyle	0xd683	tap.rgbrcInsideDefault[0] (border definition for inside horizontal borders)	BRC	variable length
sprmTCellBrcInsideVStyle	0xd684	tap.rgbrcInsideDefault[1] (border definition for inside vertical borders)	BRC	variable length
sprmTCellBrcTL2BRStyle	0xd685	tap.tcDefault.tcd.brcTL2BR (border definition for diagonal border)	BRC	variable length
sprmTCellBrcTR2BLStyle	0xd686	tap.tcDefault.tcd.brcTR2BL (border definition)	BRC	variable length
sprmTCellShdStyle	0xd687	tap.tcDefault.shd (shading definition for table style)	SHD	variable length
sprmTCHorzBands	0x3488	tap.cHorzBands (size of a horizontal style band)	count of rows	byte
sprmTCVertBands	0x3489	tap.cVertBands (size of a vertical style band)	count of columns	byte
sprmTJc	0x548a	Changes tap.jc, the justification code for the table.	row alignment	word
sprmTTableBrcTop	0xd68b	tap.rgbrcTable[ibrcTop] (default border for all cells in this row)	BRC	variable length

Name	Sprm	Property modified	Parameter	Parameter size
sprmTTableBrcLeft	0xd68c	tap.rgbrcTable[ibrcLeft] (default border for all cells in this row)	BRC	variable length
sprmTTableBrcBottom	0xd68d	tap.rgbrcTable[ibrcBottom] (default border for all cells in this row)	BRC	variable length
sprmTTableBrcRight	0xd68e	tap.rgbrcTable[ibrcRight] (default border for all cells in this row)	BRC	variable length
sprmTTableBrcInsideH	0xd68f	tap.rgbrcTable[ibrcInsideH] (default border for all cells in this row)	BRC	variable length
sprmTTableBrcInsideV	0xd690	tap.rgbrcTable[ibrcInsideV] (default border for all cells in this row)	BRC	variable length
sprmTFBiDi	0x560b	tap.fBiDi	0 or 1	word (2 bytes)
sprmTFBiDi90	0x5664	tap.fRTL	0 or 1	word (2 bytes)

Complex SPRMs

Complex Paragraph SPRMs

sprmPIstdPermute (opcode 0xC601) is a complex sprm which is applied to a piece when the style codes of paragraphs within a piece must be mapped to other style codes. It has the following format:

Field	Size	Comment
sprm	short	opcode(==0xC601)
cch	byte	Count of bytes (not including sprm and cch)
fLongg	byte	Always 0
fSpare	byte	Always 0
istdFirst	unsigned short	Index of first style in range to which permutation stored in rgistd applies
istdLast	unsigned short	Index of last style in range to which permutation stored in rgistd applies
rgistd[]	unsigned short	Array of istd entries that records the mapping of istds for text copied from a source document to istds that exist in the destination document after the text was pasted

To interpret sprmPIstdPermute, first check if pap.istd is greater than the istdFirst recorded in the sprm and less than or equal to the istdLast recorded in the sprm. If it is not, the sprm has no effect. If it is, pap.istd is set to rgistd[pap.istd-istdFirst]. sprmPIstdPermute is only stored in grpprls linked to a piece table. It should never be recorded in a PAPX.

sprmPIncLvl (opcode 0x2602) is applied to pieces in the piece table that contain paragraphs with style codes (istds) >= 1 and <= 9. These style codes identify heading levels in a Word outline structure. The sprm causes a set of paragraphs to be changed to a new heading level.

The sprm is three bytes long and consists of the sprm code and a one byte two's complement value.

If pap.stc is < 1 or > 9, sprmPIncLvl has no effect. Otherwise, if the value stored in the byte has its highest order bit off, the value is a positive difference which should be added to pap.istd and pap.lvl and then pap.stc should be set to min(pap.istd, 9). If the byte value has its highest order bit on, the value is a negative difference which should be sign extended to a word and then subtracted from pap.istd and pap.lvl. Then pap.stc should be set to max(1, pap.istd). sprmPIncLvl is only stored in grpprls linked to a piece table.

sprmPIlfo (opcode 0x460B) sets the pap.ilfo. Its argument, an ilfo, is an index into the document's hpllfo, which contains the list data for that paragraph, describing the appearance of the automatic number at the beginning of the paragraph. A value of zero means the paragraph is not numbered, and a value of 2047 indicates the paragraph came from a pre-Word 97 file so the formatting information is still stored in the pap.anld and the paragraph should be converted to Word 97 format.

sprmPIlvl (opcode (0x260A) sets the pap.ilvl. It takes an index (0 through 8) to indicate which level of a multilevel list this paragraph belongs to. For simple (one-level lists) or unnumbered paragraphs, this value should always be zero.

sprmPAnld80 (opcode 0xC63E) is currently only used for compatibility with pre-Word 97 docs. It sets the pap.anld, which before Word 97 described the automatic number at the beginning of any numbered paragraph. It is used only long enough to put the data into the document's list table (rglst) and set the pap.ilfo to point to the proper entry in the list table. The pap.anld is only relevant if pap.ilfo==2047 (see sprmPIlfo above).

The **sprmPChgTabsPapx** (opcode 0xC60D) is a complex sprm that describes changes in tab settings from the underlying style. It is only stored as part of PAPXs stored in FKPs and in the STSH. It has the following format:

Field	Size	Comment
sprm	short	opcode
cch	byte	Count of bytes (not including sprm and cch)
itbdDelMax	byte	Number of tabs to delete
rgdxaDel	int[itbdDelMax]	Array of tab positions for which tabs should be deleted
itbdAddMax	byte	Number of tabs to add
rgdxaAdd	int[itbdAddMax]	Array of tab positions for which tabs should be added
rgtbdAdd	byte[itbdAddMax]	Array of tab descriptors corresponding to rgdxaAdd

When **sprmPChgTabsPapx** is interpreted, the rgdxaDel of the sprm is applied first to the pap that is being transformed. This is done by deleting from the pap the rgdxaTab entry and rgtbd entry of any tab whose rgdxaTab value is equal to one of the rgdxaDel values in the sprm. It is guaranteed that the entries in pap.rgdxaTab and the sprm's rgdxaDel and rgdxaAdd are recorded in ascending dxa order. Then the rgdxaAdd and rgtbdAdd entries are merged into the pap's rgdxaTab and rgtbd arrays so the resulting pap rgdxaTab is sorted in ascending order with no duplicates.

sprmPNest80 (opcode 0x4610) causes its operand, a two-byte dxa value to be added to pap.dxaLeft for Word 97. If the result of the addition is less than 0, 0 is stored into pap.dxaLeft. It is used to shift the left indent of a paragraph to the right or left. sprmPNest is only stored in grpprls linked to a piece table.

sprmPNest (opcode 0x465f) is the Word 2000 version. The difference is the dxaLeft in Word 2000 is logical (it is left indent for Left-to-right text but right indent for Right-to-left text).

sprmPDyaLine (opcode 0x6412) moves a 4 byte LSPD structure into pap.lspd. Two short fields are stored in this data structure. The first short in the structure is named lspd.dyaLine and the second is named lspd.fMultLinespace. When lspd.fMultLinespace is 0, the magnitude of lspd.dyaLine specifies the amount of space provided for lines in the paragraph in twips. When lspd.dyaLine is positive, Word ensures that AT LEAST the magnitude of lspd.dyaLine is reserved on the page for each line displayed in the paragraph. If the height of a line becomes greater than lspd.dyaLine, the size calculated for that line is reserved on the page. When lspd.dyaLine is negative, Word ensures that EXACTLY the magnitude of lspd.dyaLine (-lspd.dyaLine) is reserved on the page for each line displayed in the paragraph. When lspd.fMultLinespace is 1, Word reserves for each line the (maximal height of the line*lspd.dyaLine)/240.

The **sprmPChgTabs** (opcode 0xC615) is a complex sprm which describes changes to tab settings for any paragraph within a piece. It is only stored as part of a grpprl linked to a piece table. It has the following format:

Field	Size	Comment
sprm	short	Opcode
cch	byte	Count of bytes (not including sprm and cch)
itbdDelMax	byte	Number of tabs to delete
rgdxaDel	int[itbdDelMax]	Array of tab positions for which tabs should be deleted
rgdxaClose	int[itbdDelMax]	Array of tolerances corresponding to rgdxaDel where each tolerance defines an interval around a corresponding rgdxaDel entry within which all tabs should be removed
itbdAddMax	byte	Number of tabs to add
rgdxaAdd	int[itbdAddMax]	Array of tab positions for which tabs should be added
rgtbdAdd	byte[itbdAddMax]	Array of tab descriptors corresponding to rgdxaAdd

itbdDelMax and itbdAddMax are defined to be equal to 50. This means that the largest possible instance of sprmPChgTabs is 354. When the length of the sprm is >= 255, the cch field will be set equal to 255. When cch==255, the actual length of the sprm can be calculated as follows: length=2+itbdDelMax*4+itbdAddMax*3.

When sprmPChgTabs is interpreted, the rgdxaDel of the sprm is applied first to the pap that is being transformed. This is done by deleting from the pap the rgdxaTab entry and rgtbd entry of any tab whose rgdxaTab value is within the interval [rgdxaDel[i]-rgdxaClose[i], rgdxaDel[i]+rgdxaClose[i]]. It is guaranteed that the entries in pap.rgdxaTab and the sprm's rgdxaDel and rgdxaAdd are recorded in ascending dxa order. Then the rgdxaAdd and rgtbdAdd entries are merged into the pap's rgdxaTab and rgtbd arrays so the resulting pap rgdxaTab is sorted in ascending order with no duplicates.

sprmPPc (opcode 0x261B) is a complex sprm 3 bytes long which describes changes in the pap.pcHorz and pap.pcVert. It is able to change both fields' contents in parallel. It has the following format:

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	sprm	short			Opcode

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
2	2		short	:4	F0	Reserved
		pcVert	short	:2	0C	If pcVert==3, pap.pcVert should not be changed. Otherwise, contains new value of pap.pcVert.
		pcHorz	short	:2	03	If pcHorz==3, pap.pcHorz should not be changed. Otherwise, contains new value of pap.pcHorz.

sprmPPc is interpreted by moving pcVert to pap.pcVert if pcVert!=3 and by moving pcHorz to pap.pcHorz if pcHorz!=3. sprmPPc is stored in PAPX FKPs and also in grpprls linked to piece table entries.

sprmPPropRMark (opcode 0xC63F) is interpreted by moving the first parameter byte to pap.fPropRMark, the next two bytes to pap.ibstPropRMark, and the remaining four bytes to pap.dttmPropRMark.

sprmPHugePapx is stored in PAPX FKPs in place of the grpprl of a PAPX which would otherwise be too big to fit in an FKP (as of this writing, 488 bytes is the size of the largest PAPX which can fit in an FKP). The parameter fc gives the location of the grpprl in the data stream. The first word at that fc counts the number of bytes in the grpprl (not including the byte count itself). A sprmPHugePapx should therefore only be found in a PAPX FKP and should be the only sprm in that PAPX's grpprl.

Complex Character SPRMs

sprmCPicLocation (opcode 0x6A03) is used ONLY IN CHPX FKPs. This sprm moves the 4-byte operand of the sprm into the chp.fcPic field. It simultaneously sets chp.fSpec to 1. This sprm is also used when the chp.lTagObj field that is unioned with chp.fcPic is to be set for OLE objects.

sprmCChs (opcode 0xEA08) is used to record a character set id for text that was pasted into the Word document that used a character set different than Word's default character set. When chp.fChsDiff==0, the character set used for a run of text is the default character set for the version of Word that last saved the document. When chp.fChsDiff==1, chp.chse specifies the character set used for this run of text. This sprm's operand is 3 bytes. When this sprm is interpreted, the first byte of the operand is moved to chp.fChsDiff and the remaining word is moved to chp.chse.

sprmCSymbol (opcode 0x6A09) is used to specify the font and the character used within that font to display a symbol character in Word. This sprm's operand is 4 bytes. The first 2 bytes hold the font code; the last 2 bytes hold a character specifier. When this sprm is interpreted, the font code is moved to chp.ftcSym and the character specifier is moved to chp.xchSym and chp.fSpec is set to 1.

sprmCIstdPermute (opcode 0xCA31) (which has the same format as sprmPIstdPermute (opcode 0xC601)) is a complex sprm which is applied to a piece when the style codes for character styles tagging character runs within a piece must be mapped to other style codes. This is the same format as sprmPIstdPermute (opcode 0xC601). It has the following format:

Field	Size	Comment
sprm	short	opcode(==0xCA31)
cch	byte	Count of bytes (not including sprm and cch)
fLongg	byte	Always 0
fSpare	byte	Always 0

Field	Size	Comment
istdFirst	unsigned short	Index of first style in range to which permutation stored in rgistd applies
istdLast	unsigned short	Index of last style in range to which permutation stored in rgistd applies
rgistd[]	unsigned short	Array of istd entries that records the mapping of istds for text copied from a source document to istds that exist in the destination document after the text was pasted

To interpret sprmCIstdPermute, first check if chp.istd is greater than the istdFirst recorded in the sprm and less than or equal to the istdLast recorded in the sprm. If it is not, the sprm has no effect. If it is, chp.istd is set to rgstd[chp.istd – istdFirst] and any chpx stored in that rgstd entry is applied to the chp. sprmCIstdPermute is only stored in grpprls linked to a piece table. It should never be recorded in a CHPX.

Note: it is possible an istd may be recorded in the rgistd that refers to a paragraph style. This has no harmful consequences since the istd for a paragraph style should never be recorded in chp.istd.

sprmCDefault (opcode 0x2A32) clears the fBold, fItalic, fOutline, fStrike, fShadow, fSmallCaps, fCaps, fVanish, kul and ico fields of the chp to 0. It was first defined for Word 3.01 and had to be backward compatible with Word 3.00 so it is a variable length sprm whose count of bytes is 0. It consists of the sprmCDefault opcode followed by a byte of 0. sprmCDefault is stored only in grpprls linked to piece table entries.

sprmCPlain (opcode 0x2A33) is used to make the character properties of runs of text equal to the style character properties of the paragraph that contains the text. When Word interprets this sprm, the style sheet CHP is copied over the original CHP preserving the fSpec setting from the original CHP. sprmCPlain is stored only in grpprls linked to piece table entries.

sprms 0x0835 through 0x083C (sprmCFBold through sprmCFVanish) set single bit properties in the CHP. When the parameter of the sprm is set to 0 or 1, then the CHP property is set to the parameter value.

When the parameter of the sprm is 128, then the CHP property is set to the value that is stored for the property in the style sheet CHP. When the parameter of the sprm is 129, the CHP property is set to the negation of the value that is stored for the property in the style sheet CHP. sprmCFBold through sprmCFVanish are stored only in grpprls linked to piece table entries.

sprmCSizePos (opcode 0xEA3F) is a five-byte sprm consisting of the sprm opcode and a three byte parameter. The sprm has the following format:

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	sprm	short			Opcode
2	2	hpsSize	short	:8	FF	When != 0, contains new size of chp.hps
3	3	cInc	short	:7	FE	Contains the number of font levels to increase or decrease size of chp.hps as a two's complement value.
		fAdjust	short	:1	01	When ==1, means that chp.hps should be adjusted up or down by one font level for super/subscripting change
4	4	hpsPos	short	:8	FF	When !=128, contains super/subscript position as a two's complement number

When Word interprets this sprm, if hpsSize != 0 then chp.hps is set to hpsSize. If cInc != 0, the cInc is interpreted as a 7 bit two's complement number and the procedure described

below for interpreting sprmCHpsInc is followed to increase or decrease the chp.hps by the specified number of levels. If hpsPos!=128, then chp.hpsPos is set equal to hpsPos. If fAdjust is on , hpsPos!=128 and hpsPos!=0 and the previous value of chp.hpsPos==0, then chp.hps is reduced by one level following the method described for sprmCHpsInc. If fAdjust is on, hpsPos==0 and the previous value of chp.hpsPos!=0, then the chp.hps value is increased by one level using the method described below for sprmCHpsInc.

sprmCHpsInc(opcode 0x2A44) is a three-byte sprm consisting of the sprm opcode and a one-byte parameter. Word keeps an ordered array of the font sizes that are defined for the fonts recorded in the system file with each font size transformed into an hps. The parameter is a one-byte two's complement number. Word uses this number to calculate an index in the font size array to determine the new hps for a run. When Word interprets this sprm and the parameter is positive, it searches the array of font sizes to find the index of the smallest entry in the font size table that is greater than the current chp.hps. It then adds the parameter minus 1 (-1) to the index and maxes this with the index of the last array entry. It uses the result as an index into the font size array and assigns that entry of the array to chp.hps.

When the parameter is negative, Word searches the array of font sizes to find the index of the entry that is less than or equal to the current chp.hps. It then adds the negative parameter to the index and does a min of the result with 0. The result of the min function is used as an index into the font size array and that entry of the array is assigned to chp.hps. sprmCHpsInc is stored only in grpprls linked to piece table entries.

sprmCHpsPosAdj (opcode 0x2A46) causes the hps of a run to be reduced the first time text is superscripted or subscripted and causes the hps of a run to be increased when superscripting/subscripting is removed from a run. The one byte parameter of this sprm is the new hpsPos value to be stored in chp.hpsPos. If hpsPos!=0 (meaning that the text is to be super/subscripted), Word first examines the current value of chp.hpsPos to see if it is equal to 0. If so, Word uses the algorithm described for sprmCHpsInc to decrease chp.hps by one level. If the new hpsPos==0 (meaning the text is not super/subscripted), Word examines the current chp.hpsPos to see if it is not equal to 0. If it is not (which means text is being restored to normal position), Word uses the sprmCHpsInc algorithm to increase chp.hps by one level. After chp.hps is adjusted, the parameter value is stored in chp.hpsPos. sprmCHpsPosAdj is stored only in grpprls linked to piece table entries.

The parameter of **sprmCMajority** (opcode 0xCA47) is itself a list of character sprms which encodes a criterion under which certain fields of the chp are to be set equal to the values stored in a style's CHP. Bytes 0 and 1 of sprmCMajority contains the opcode, byte 2 contains the length of the following list of character sprms. Word begins interpretation of this sprm by applying the stored character sprm list to a standard chp. That chp has chp.istd=istdNormalChar. chp.hps=20, chp.lid=0x0400, and chp.ftc=4. Word then compares fBold, fItalic, fStrike, fOutline, fShadow, fSmallCaps, fCaps, ftc, hps, hpsPos, kul, cv, and ico in the original CHP with the values recorded for these fields in the generated CHP. If a field in the original CHP has the same value as the field stored in the generated CHP, then that field is reset to the value stored in the style's CHP. If the two copies differ, then the original CHP value is left unchanged. sprmCMajority is stored only in grpprls linked to piece table entries.

sprmCHpsInc1 (opcode 0xCA4A) is used to increase or decrease chp.hps by increments of 1. This sprm is interpreted by adding the two byte increment stored as the opcode of the sprm to chp.hps. If this result is less than 8, the chp.hps is set to 8. If the result is greater than 32766, the chp.hps is set to 32766.

sprmCMajority50 (opcode 0xCA4C) has the same format as sprmCMajority and is interpreted in the same way.

sprmCPropRMark (opcode 0xCA57) is interpreted by moving the first parameter byte to `chp.fPropRMark`, the next two bytes to `chp.ibstPropRMark`, and the remaining four bytes to `chp.dttmPropRMark`.

sprmCDispFldRMark (opcode 0xCA62) is interpreted by moving the first parameter byte to `chp.fDispFldRMark`, the next two bytes to `chp.ibstDispFldRMark`, the next four bytes to `chp.dttmDispFldRMark`, and the remaining 32 bytes to `chp.xstDispFldRMark`.

sprmCRgftc0 (opcode 0x4A4F), **sprmcCRgftc1**(opcode 0x4A50), and **sprmCRgftc2** (opcode 0x4A4F) are used to specify the available fonts for use with text. `Rgftc0` specifies the font used for characters U+0000 through U+007F. `Rgftc1` specifies the font to use for East Asian characters, and `Rgftc2` specifies the font to use for all other text. See Appendix B for details on how the font is calculated.

sprmCRglid0 (opcode 0x486D) and **sprmCRglid1** (opcode 0x486E) are used to specify the languages available for use with the text in this run. `sprmCRglid1` specifies the language for East Asian text, `sprmCRglid0` specifies the language for all other text. See Appendix B for details on how the language is calculated.

sprmCIdctHint (opcode 0x286F) specifies a script bias for the text in the run. For Unicode characters shared between East Asian and non-East Asian scripts, this property determines what font and language the character will use. When this value is 0, text properties bias towards non-East Asian properties. When this value is 1, text properties bias towards East Asian properties. See Appendix B for details on the calculation of font and language properties.

sprmCCvPermute (0xca7c) follows the permute structure: `SPRM+size+Old Value+New Value`. The new value is applied if the current value equals the old value. (So, `sprmCCvPermute+Red+Blue` is: if `(pchp->cv==Red)` then `(pchp->cv==Blue)`. This allows us to set a value if the user hasn't changed it since we wanted to permute it, so if the user changed the color to yellow after we decided to change Red to Blue, we would leave the users choice of Yellow alone.)

Complex Picture SPRMs

sprmPicScale (opcode 0xCE01) is used to scale the x and y dimensions of a Word picture and to set the cropping for each side of the picture. The `sprm` begins with the two-byte opcode, followed by the length of the parameter (always 12) stored in a byte. The 12-byte long operand consists of an array of 6 two-byte integer fields. The 0th integer contains the new setting for `pic.mx`. The 1st integer contains the new setting for `pic.my`. The 2nd integer contains the new setting for `pic.dxaCropLeft`. The 3rd integer contains the new setting for `pic.dyaCropTop`. The 4th integer contains the new setting for `pic.dxaCropRight`. The 5th integer contains the new setting of `pic.dxaCropBottom`. `sprmPicScale` is stored only in `grpprl`s linked to piece table entries.

Complex Section SPRMs

sprmSPropRMark (opcode 0xD227) is interpreted by moving the first parameter byte to `sep.fPropRMar`k, the next two bytes to `sep.ibstPropRMark`, and the remaining four bytes to `sep.dttmPropRMark`.

sprmSTextFlow (opcode 0x5033) represents the text flow to be applied to this section. Possible values are:

0 Horizontal, **non-@font**

1 Top to bottom, @font

2 Bottom to top, **non-@font**

3 Top to bottom, **non-@font**

4 Horizontal, @-font

Complex Table SPRMs

sprmTDxaLeft (opcode 0x9601) is called to adjust the x position within a column which marks the left boundary of text within the first cell of a table row. This sprm causes a whole table row to be shifted left or right within its column leaving the horizontal width and vertical height of cells in the row unchanged. Bytes 0-1 of the sprm contain the opcode, and the new dxa position, named dxaNew, is stored as an integer in bytes 2 and 3. Word interprets this sprm by adding dxaNew-(rgdxaCenter[0]+tap.dxaGapHalf) to every entry of tap.rgdxaCenter whose index is less than tap.itcMac. sprmTDxaLeft is stored only in grpprls linked to piece table entries.

sprmTDxaGapHalf (opcode 0x9602) adjusts the white space that is maintained between columns by changing tap.dxaGapHalf. Because the left boundary of text within the leftmost cell should be at the same location after the sprm is applied, Word also adjusts tap.rgdxCenter[0] by the amount that tap.dxaGapHalf changes. Bytes 0-1 of the sprm contain the opcode, and the new dxaGapHalf, named dxaGapHalfNew, is stored in bytes 2 and 3. When the sprm is interpreted, the change between the old and new dxaGapHalf values, tap.dxaGapHalf-dxaGapHalfNew, is added to tap.rgdxaCenter[0] and then dxaGapHalfNew is moved to tap.dxaGapHalf. sprmTDxaGapHalf is stored in PAPXs and also in grpprls linked to piece table entries.

sprmTTableBorders80 (opcode 0xD605) sets the tap.rgbrcTable. The sprm is interpreted by moving the 24 bytes of the sprm's operand to tap.rgbrcTable.

sprmTTableBorders (opcode 0xD613) sets the tap.rgbrcTable. The sprm is interpreted by moving the 48 bytes of the sprm's operand (array of BRCs) to tap.rgbrcTable.

sprmTDefTable10 (opcode0xD606) is an obsolete version of sprmTDefTable (opcode 0xD608) that was used in WinWord 1.x. Its contents are identical to those in sprmTDefTable, except that the TC structures contain the obsolete structures BRC10s.

sprmTDefTable (opcode 0xD608) defines the boundaries of table cells (tap.rgdxaCenter) and the properties of each cell in a table (tap.rgtc) for Word 97. Bytes 0 and 1 of the sprm contain its opcode. Bytes 2 and 3 store a two-byte length of the following parameter. Byte 4 contains the number of cells to be defined by the sprm, named itcMac. When the sprm is interpreted, itcMac is moved to tap.itcMac. itcMac cannot be larger than 32. In bytes 5 through 5+2*(itcMac+1)-1, is stored in an array of integer dxa values sorted in ascending order which are moved to tap.rgdxaCenter. In bytes 5+2*(itcMac+1) through byte 5+2*(itcMac+1)+10*itcMac-1 is stored in an array of TC entries corresponding to the stored tap.rgdxaCenter. This array is moved to tap.rgtc. sprmTDefTable is only stored in PAPXs.

sprmTDefTableShd80 (opcode 0xD609) is similar to sprmTDefTable, and complements it by defining the shading of each cell in a table (tap.rgshd) for Word 97. Bytes 0 and 1 of the sprm contain its opcode. Bytes 2 and 3 store a two-byte length of the following parameter. Byte 4 contains the number of cells to be defined by the sprm, named itcMac. itcMac cannot be larger than 32. In bytes 5 through 5+2*(itcMac+1)-1, is stored an array of SHD80s. This array is moved to tap.rgshd. sprmTDefTable80 is only stored in PAPXs.

sprmTDefTableShd (opcode 0xd612). Operates on tap.rgtc[].shd. The opcode is followed by the byte size in a short ((number of SHDs)*sizeof(SHD)), then an array of SHDs. It only operates on columns 0 to 21 because anything larger would overflow the variable sprm length of 255.

sprmTDefTableShd2nd (opcode 0xd616). Same as SprmTDefTableShd but for columns 22-43.

sprmTDefTableShd3rd (opcode 0xd60c). Same as SprmTDefTableShd but for columns 44-63.

SprmTSetBrc80 (opcode 0xD620) allows the border definitions (BRC80s) within TCs to be set to new values. It has the following format:

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	sprm	short			opcode 0xD620
2	2	count	byte			Number of bytes for operand
3	3	itcFirst	byte			The index of the first cell that is to have its borders changed.
4	4	itcLim	byte			Index of the cell that follows the last cell to have its borders changed
5	5		short	:4	F0	Reserved
		fChangeRight	short	:1	08	=1 when tap.rgtc[].brcRight is to be changed
		fChangeBottom	short	:1	04	=1 when tap.rgtc[].brcBottom is to be changed
		fChangeLeft	short	:1	02	=1 when tap.rgtc[].brcLeft is to be changed
		fChangeTop	short	:1	01	=1 when tap.rgtc[].brcTop is to be changed
6	6	brc	BRC80			New BRC value to be stored in TCs.

This sprm changes the brc fields selected by the fChange* flags in the sprm to the brc value stored in the sprm, for every tap.rgtc entry whose index is greater than or equal to itcFirst and less than itcLim. sprmTSetBrc is stored only in grpprls linked to piece table entries.

sprmTSetBrc (opcode 0xD62F) works in the same manner as sprmTSetBrc80 but uses the new BRC structure introduced in 2000.

sprmTInsert (opcode 0x7621) inserts new cell definitions in an existing table's cell structure. Bytes 0 and 1 of the sprm contain the opcode. Byte 2 is the index within tap.rgdxaCenter and tap.rgtc at which the new dxaCenter and tc values are inserted. This index is named itcInsert. Byte 3 contains a count of the cell definitions to add to the tap, named ctc. Bytes 4 and 5 contain the width of the cells to add, named dxaCol. If there are already cells defined at the index where cells are to be inserted, tap.rgdxaCenter entries at or above this index must move to the entry ctc higher and must be adjusted by adding ctc*dxaCol to the value stored. The contents of tap.rgtc at or above the index must be moved 10*ctc bytes higher in tap.rgtc. If itcInsert is greater than the original tap.itcMac, itcInsert – tap.ctc columns beginning with index tap.itcMac must be added of width dxaCol (loop from itcMac to itcMac+itcInsert-tap.ctc adding dxaCol to the rgdxaCenter value of the previous entry and storing the sum as dxaCenter of the new entry), whose TC entries are cleared to zeros. Beginning with index itcInsert, ctc columns of width dxaCol must be added by constructing new tap.rgdxaCenter and tap.rgtc entries with the newly defined rgtc entries cleared to zeros. Finally, the sum of the number of cells added to the tap is added to tap.itcMac. sprmTInsert is stored only in grpprls linked to piece table entries.

sprmTDelete (opcode 0x5622) deletes cell definitions from an existing table's cell structure. Bytes 0 and 1 of the sprm contain the opcode. Byte 2 contains the index of the first cell to delete, named itcFirst. Byte 3 contains the index of the cell that follows the last cell to be deleted, named itcLim. sprmTDelete causes any rgdxaCenter and rgtc entries whose index is greater than or equal to itcLim to move to the entry that is itcLim-itcFirst

lower, and causes tap.itcMac to decrease by the number of cells deleted. sprmTDelete is stored only in grpprls linked to piece table entries.

sprmTDxaCol (opcode 0x7623) changes the width of cells whose index is within a certain range to be a certain value. Bytes 0 and 1 of the sprm contain the opcode. Byte 2 contains the index of the first cell whose width is to change, named itcFirst. Byte 3 contains the index of the cell that follows the last cell whose width is to change, named itcLim. Bytes 4 and 5 contain the new width of the cell, named dxaCol. This sprm causes the itcLim-itcFirst entries of tap.rgdxaCenter to be adjusted so tap.rgdxaCenter[i+1] = tap.rgdxaCenter[i]+dxaCol. Any tap.rgdxaCenter entries that exist beyond itcLim are adjusted to take into account the amount added to or removed from the previous columns. sprmTDxaCol is stored only in grpprls linked to piece table entries.

sprmTMerge (opcode 0x5624) merges the display areas of cells within a specified range. Bytes 0 and 1 of the sprm contain the opcode. Byte 2 contains the index of the first cell to merge, named itcFirst. Byte 3 contains the index of the cell that follows the last cell to merge, named itcLim. This sprm causes tap.rgtc[itcFirst].fFirstMerged to be set to 1. Cells in the range whose index is greater than itcFirst and less than itcLim have tap.rgtc[].fMerged set to 1. sprmTMerge is stored only in grpprls linked to piece table entries.

sprmTSplit (opcode 0x5625) splits the display areas of merged cells into their originally assigned display areas. Bytes 0 and 1 of the sprm contain the opcode. Byte 2 contains the index of the first cell to split, named itcFirst. Byte 3 contains the index of the cell that follows the last cell to split, named itcLim. This sprm clears tap.rgtc[].fFirstMerged and tap.rgtc[].fMerged for all rgtc entries >= itcFirst and < itcLim. sprmTSplit is stored only in grpprls linked to piece table entries.

sprmTSetBrc10 (opcode 0xD626) has the same format as sprmTSetBrc but uses the old BRC10 structure.

sprmTSetShd80 (opcode 0x7627) allows the Word 97 style shading definitions (SHD80s) within a tap to be set to new values. Bytes 0 and 1 of the sprm contain the opcode. Byte 2 contains the index of the first cell whose shading is to change, named itcFirst. Byte 3 contains the index of the cell that follows the last cell whose shading is to change, named itcLim. Bytes 4 and 5 contain the SHD80 structure, named shd80. This sprm causes the itcLim-itcFirst entries of tap.rgshd to be set to shd. sprmTSetShd is stored only in grpprls linked to piece table entries.

sprmTSetShdOdd80 (opcode 0x7628) is identical to sprmTSetShd80, but it only changes the rgshd for odd indices between itcFirst and itcLim. sprmTSetShdOdd80 is stored only in grpprls linked to piece table entries.

sprmTSetShd (opcode 0xd62d) is identical to sprmTSetShd80 but uses the Word 2000 style shading structure (SHD) and changes shading structures in tap.rgtc[].shd (so the indices are indices into the rgtc).

sprmTSetShdOdd (opcode 0xd62e) is identical to sprmTSetShd, but it only changes the rgshd for odd indices between itcFirst and itcLim.

sprmTVertMerge (opcode 0xD62B) changes the vertical cell merge properties for a cell in the tap.rgtc[]. Bytes 0 and 1 of the sprm contain the opcode. Byte 2 contains the index of the cell whose vertical cell merge properties are to change. Byte 3 sets the new vertical cell merge properties for the cell, a 0 clears both fVertMerge and fVertRestart, a 1 sets fVertMerge and clears fVertRestart, and a 3 sets both flags. sprmTVertMerge is stored only in grpprls linked to piece table entries.

sprmTVertAlign (opcode 0xD62C) changes the vertical alignment property in the tap.rgtc[]. Bytes 0 and 1 of the sprm contain the opcode. Byte 2 contains the index of the

first cell whose shading is to change, named itcFirst. Byte 3 contains the index of the cell that follows the last cell whose shading is to change, named itcLim. This sprm causes the vertAlign properties of the itcLim–itcFirst entries of tap.rgtc[] to be set to the new vertical alignment property contained in Byte 4. sprmTVertAlign is stored only in grpprls linked to piece table entries.

sprmTCellPadding (0xd632), **sprmTCellPaddingDefault** (0xd634), **sprmTCellPaddingOuter** (0xd638), **sprmTCellSpacing** (0xd631), **sprmTCellSpacingDefault** (0xd633), **sprmTCellSpacingOuter** (0xd637) all have the same parameter, a CSSA, which always has a length of 6 (size stored in the first byte).

A CSSA looks like this:

```
struct {
uchar itcFirst;
uchar itcLim
uchar grfbrc;
uchar ftsWidth;
short wWidth;
} CSSA;
```

The itcFirst and itcLim specify the indexes, respectively, of the first cell affected by the SPRM and the first cell NOT affected by the SPRM, counting from 0. So if they are 1 and 3, that means the 2nd and 3rd cell are affected, but not the 4th (index 3) or 1st (index 0). For the "Default" sprms, the itcs are always 0 and 1, and affect the entire table.

The grfbrc is a bit field that specifies which borders of the affected cells are affected: 0x01 means top, 0x02 means Left, 0x04 means Bottom, and 0x08 means Right.

sprmTCellWidth (0xd635) has the following parameter (size stored in the first byte):

```
uchar itcFirst;
uchar itcLim
uchar ftsWidth;
short wWidth;
```

The ftsWidth and wWidth specify the desired width and width units for all cells with index >= itcFirst and < itcLim.

sprmTTableWidth (0xf614), **sprmTWidthAfter** (0xf618), **sprmTWidthBefore** (0xf617), **sprmTWidthIndent** (0xf661) all change an ftsWidth and a wWidth value. The first byte is the ftsWidth and the remaining two bytes are the wWidth value.

sprmTPc (0x360d) behaves just like sprmPPc, but it applies to absolutely positioned tables.

sprmTDiagLine80 (0xd62a). Contains 2 BRC80s for the two diagonal lines for each column.

sprmTDiagLine (0xd630) is the same as sprmTDiagLine80, but with 2 BRC80s instead.

sprmTBrcBottomCv (0xd61c) contains the size of the parameter and an array of itcMax (64) COLORREFS, one for each TC.rgbrc[ibrcBottom].cv in tap.rgtc[itcMax].rgbrc[ibrcBottom].cv.

sprmTBrcLeftCv (0xd61b). Same as sprmTBrcBottomCv, but changing cv for rgtc[itcMax].rgbrc[ibrcLeft].

sprmTBrcRightCv (0xd61d) Same as sprmTBrcBottomCv, but changing cv for rgtc[itcMax].rgbrc[ibrcRight].

sprmTBrcTopCv (0xd61a) Same as sprmTBrcBottomCv, but changing cv for rgtc[itcMax].rgbrc[ibrcTop].

sprmTCellBrcType (0xd662) contains the brcType for the entire tap.rgtc in the following sequence: rgtc[0].brcTop.brcType, rgtc[0].brcLeft.brcType, rgtc[0].brcBottom.brcType, rgtc[0].brcRight.brcType, rgtc[1].... The first two bytes following the opcode contain the size of the parameter.

Complex File Format

There are some differences between the file format of a full saved document and that of a fast saved document. In Word 95 and earlier versions , one of the differences was the necessity to include the "complex" table information. In Word 97 and later versions, the fcClx always indicates the location of the "complex" table information and is used to determine the location and contents of text and properties. This is necessary due to the effects of Unicode and Unicode compression.

fcClx is the fc where the complex part of the file begins, and cbClx is the size (in bytes) of the complex part. The complex part of the file contains a group of grpprls that encode formatting changes made by the user and a piece table (plcfpcd). The piece table is needed because the text of the document is not stored contiguously in the file after a fast save.

The complex part of a file (CLX) is composed of a number of variable-sized blocks of data. Recorded first are any grpprls that may be referenced by the plcfpcd (if the plcfpcd has no grpprl references, no grpprls are recorded) followed by the plcfpcd. Each block in the complex part is prefaced by a clxt (**clx t**ype), which is a 1-byte code, either 1 (meaning the block contains a grpprl) or 2 (meaning this is the plcfpcd). A clxtGrpprl(1) is followed by a 2-byte cb which is the count of bytes of the grpprl. A clxtPlcfpcd(2) is followed by a 4-byte lcb which is the count of bytes of the piece table. A full saved file has no clxtGrpprls. So the formats of the two types of blocks are:

clxt = 1	clxtGrpprl
cb	Count of bytes in grpprl
grpprl	See "**Definitions**" for a description of grpprl. A grpprl can contain sprms modifying character, paragraph, table, section or picture properties.

or

clxt = 2	clxtPlcfpcd
lcb	Count of bytes in piece table
plcfpcd	Piece table

The entire CLX would look like this, depending on the number of grpprls:

```
clxtGrpprl
cb
grpprl (0th grpprl)
clxtGrpprl
cb
grpprl (1st grpprl)
...
clxtPlcfpcd
cb
plcfpcd
```

When the prm in pcds stored in the plcfpcd, contain an igrpprl (index to a grpprl), the index stored is the order in which that grpprl was stored in the CLX.

Algorithm to determine the BOUNDS OF A PARAGRAPH containing a certain character in a complex file

When a document is recorded in non-complex format, the bounds of the paragraph that contains a particular character can be found by calculating the FC coordinate of the character, searching the bin table to find an FKP page that describes that FC, fetching that FKP, and then searching the FKP to find the interval in the rgfc that encloses the character. The bounds of the interval are the fcFirst and fcLim of the paragraph with the found character. Every character >= fcFirst and < fcLim are part of the paragraph.

When a document is recorded in complex format, a piece that was originally part of one paragraph can be copied or moved to a different paragraph. To find the *beginning* of a paragraph with a desired character in a complex document, it's necessary to search for the piece containing the character in the piece table. Then calculate the FC in the file that stores the character from the piece table information. Using the FC, search the FCs FKP for the largest FC less than the character's FC, named fcTest. If the character at fcTest-1 is contained in the current piece, then the character corresponding to that FC in the piece is the first character of the paragraph. If that FC is before, or marks, the beginning of the piece, scan a piece at a time towards the beginning of the piece table until a piece is found that contains a paragraph mark. This can be done by using the end of the piece FC, finding the largest FC in its FKP that is less than or equal to the end of piece FC, and checking to see if the character in front of the FKP FC (which must mark a paragraph end) is within the piece. When such an FKP FC is found, the FC marks the first byte of paragraph text.

To find the *end* of a paragraph for a character in a complex format file, it is necessary to know the piece that contains the character and the FC assigned to the character. Using the FC of the character, search the FKP that describes the character to find the smallest FC in the rgfc that is larger than the character FC. If the FC found in the FKP is less than or equal to the limit FC of the piece, the end of the paragraph that contains the character is at the FKP FC minus 1. If the FKP FC that was found was greater than the FC of the end of the piece, scan piece by piece toward the end of the document until a piece is found that contains a paragraph end mark. It's possible to check if a piece contains a paragraph mark by using the FC of the beginning of the piece to search in the FKPs for the smallest FC in the FKP rgfc that is greater than the FC of the beginning of the piece. If the FC found is less than or equal to the limit FC of the piece, then the character that ends the paragraph is the character immediately before the FKP FC.

A special procedure must be followed to locate the last paragraph of the main document text when footnote or header/footer text is saved in a Word file (i.e. when fib.ccpFtn != 0 or fib.ccpHdr != 0).

In this case the CP of that paragraph mark is fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpMcr+fib.ccpAtn and the limit CP of the entire plcfpcd is fib.ccpText+fib.ccpFtn+fib.ccpHdr+fib.ccpMcr+fib.ccpAtn+1.

Algorithm to determine PARAGRAPH PROPERTIES for a paragraph in a complex file

Having found the index **i** of the FC in an FKP that marks the character stored in the file immediately after the paragraph's paragraph mark, use the word offset stored in the first byte of the fkp.rgbx[i-1] to find the PAPX for the paragraph. Using papx.istd to index into the properties stored for the style sheet, the paragraph properties of the style are copied to a local PAP. Then the grpprl stored in the PAPX is applied to the local PAP, and papx.istd along

with fkp.rgbx.phe are moved into the local PAP. The process thus far has created a PAP that describes what the paragraph properties of the paragraph were at the last full save. Now apply any paragraph sprms that were linked to the piece that contains the paragraph's paragraph mark. If pcd.prm.fComplex==0, pcd.prm contains 1 sprm which should only be applied to the local PAP if it is a paragraph sprm. If pcd.prm.fComplex==1, pcd.prm.igrpprl is the index of a grpprl in the CLX. If that grpprl contains any paragraph sprms, they should be applied to the local PAP. After applying all of the sprms for the piece, the local PAP contains the correct paragraph property values.

Algorithm to determine TABLE PROPERTIES for a table row in a complex file

To determine the table properties for a table row in a complex file, scan paragraph by paragraph toward the end of the table row, until a paragraph is found with pap.fTtp=1. This paragraph consists of a single row end character. This row end character is linked to the table properties of the row. To create the TAP for the table row, clear a local TAP to zeros. Then the PAPX for the row end character must be fetched from an FKP, and the table sprms that are stored in this PAPX must be applied to the local TAP. This process has created a TAP that describes what the table properties of the table row were at the last full save. Now apply any table sprms linked to the piece that contains the table row's row end character. If pcd.prm.fComplex==0, pcd.prm contains 1 sprm which should be applied to the local TAP if it is a table sprm. If pcd.prm.fComplex==1, pcd.prm.igrpprl is the index of a grpprl in the CLX. If that grpprl contains any table sprms, apply them to the local TAP. After all of the sprms for the piece are applied, the local TAP contains the correct table property values for the table row.

Note: inside nested tables (pap.itap > 1) the **end of cell** and **end of row** markers are indicated by sprmPFInnerTable Cell and sprmPFInnerTtp. They are otherwise emitted as regular paragraphs for backward compatibility. Nested tables are possible in Word 2000 and later versions.

Algorithm to determine the CHARACTER PROPERTIES of a character in a complex file

It is necessary to fetch the paragraph properties of the paragraph that contains the character. The pap.istd of the fetched properties specifies which style sheet entry provides the default character properties for the character. The character properties recorded in the style sheet for that style are copied into a local CHP. Then, the piece containing the character is located in the piece table (plcfpcd) and the fc of the character is calculated. Using the character's FC, the page number of the CHPX FKP that describes the character is found by searching the bin table (hplcfbteChpx). The CHPX FKP stored in that page is fetched and then the rgfc in the FKP is searched to locate the bounds of the run of exception text that encompasses the character. The CHPX for that run is then located within the FKP, and the CHPX is applied to the contents of the local CHP. This process creates a CHP that describes the character properties as of the last full save. Now apply any character sprms linked to the piece that contains the character. If pcd.prm.fComplex==0, pcd.prm contains 1 sprm which should be applied to the local CHP if it is a character sprm. If pcd.prm.fComplex==1, pcd.prm.igrpprl is the index of a grpprl in the CLX. If that grpprl contains any character sprms, apply them to the local CHP. After applying all of the sprms for the piece, the local CHP contains the correct properties for the character.

Characters within the same piece, same paragraph, and same run of exception text are guaranteed to have the same properties. This fact can be used to construct a scanner that can return the limit CPs and properties of a sequence of characters that all have the same properties.

Algorithm to determine the SECTION PROPERTIES of a section in a complex file

To determine which section a character belongs to and what its section properties are, it is necessary to use the CP of the character to search the plcfsed for the index **i** of the largest CP that is less than or equal to the character's CP. plcfsed.rgcp[i] is the CP of the first character of the section and plcfsed.rgcp[i+1] is the CP of the character following the section mark that terminates the section (named cpLim). Then retrieve plcfsed.rgsed[i]. The FC in this SED gives the location where the SEPX for the section is stored. Then create a local SEP with default section properties. If the sed.fc!=0xFFFFFFFF, then the sprms within the SEPX that is stored at offset sed.fc must be applied to the local SEP. This process creates a SEP that describes what the section properties of the section were at the last full save. Now apply any section sprms that were linked to the piece that contains the section's section mark. If pcd.prm.fComplex==0, pcd.prm contains 1 sprm which should be applied to the local SEP if it is a section sprm. If pcd.prm.fComplex==1, pcd.prm.igrpprl is the index of a grpprl in the CLX. If that grpprl contains any section sprms, they should be applied to the local SEP. After applying all of the section sprms for the piece, the local SEP contains the correct section properties.

Algorithm to determine the PIC of a picture in a complex file.

The picture sprms contained in the prm's grpprl apply to any picture characters within the piece that have their chp.fSpec character ==fTrue. The picture properties for a picture (the PIC described in the Structure Definitions) are derived by fetching the PIC stored with the picture and applying to that PIC any picture sprms linked to the piece containing the picture special character.

Footnotes & Endnotes

In Word the text of footnotes and endnotes is anchored to a particular position within the document's main text, the location of its footnote/endnote reference. The following discussion only describes footnotes, with endnotes being handled identically, except that the endnote data structures contain the "edn" abbreviation where footnote data structures contain the "fnd" abbreviation. There is a structure referenced by the fib, the plcffndRef, which records the locations of the footnote references within the main text address space and another structure referenced by the fib, the plcffndTxt, which records the beginning locations of corresponding footnote text within the footnote text address space. The footnote text characters in a full saved file begin at offset fib.fcMin+fib.ccpText and extends till fib.fcMin+fib.ccpText+fib.ccpFtn. In a complex fast-saved document, the footnote text begins at CP fib.ccpText and extends till fib.ccpText+fib.ccpFtn. To find the location of the **ith** footnote reference in the main text address space, look up the **ith** entry in the plcffndRef and find the location of the text corresponding to the reference within the footnote text address space by looking up the **ith** entry in the plcffndTxt.

When there are **n** footnotes, the plcffndTxt structure consists of **n+2** CP entries. The CP entries mark the beginning character position within the footnote text address space of the footnote text for the footnotes defined for the file. The beginning CP of the text of the **ith** footnote is the **ith** CP within the plcffndTxt. The limit CP of the text of the **ith** footnote is the **i+1st** CP within the plcffndTxt.

The last character of footnote text for a footnote (i.e. the character at limit CP-1) is always a paragraph end (ASCII 13). If there are **n** footnotes, the **n+2nd** CP entry value is always 1 greater than the **n+1st** CP entry value. A paragraph end (ASCII 13) is always stored at the file position marked by the **n+1st** CP value.

When there are **n** footnotes, the plcffndRef structure consists of **n+1** CP entries followed by **n** integer flags, named fAuto. The **ith** CP in the plcffndRef corresponds to the **ith** fAuto flag. The CP entries give the locations of footnote references within the main text address space. The **n+1th** CP entry contains the value fib.ccpText+fib.ccpFtn+fib.ccpHdr+1. The fAuto flag contains 1 whenever the footnote reference name is auto-generated by Word.

When a footnote reference name is automatically generated by Word, Word generates the name by adding 1 to the index number of the reference in the plcffndRef and translating that number to ASCII text. When the footnote reference is auto generated, the character at the main text CP position for the footnote reference should be a footnote reference character (ASCII 5) which has a chp recorded with chp.fSpec=1.

The number of footnotes stored in a Word binary file can be found by (fib.cbPlcffndTxt/4)-1.

Headers and Footers

The header and footer text characters in a full saved file begin at offset fib.fcMin+fib.ccpText+fib.ccpFtn and extend till fib.fcMin+fib.ccpText+fib.ccpFtn+fib.ccpHdr. In a complex fast-saved document, the footnote text begins at CP fib.ccpText+fib.ccpFtn and extends till fib.ccpText+fib.ccpFtn+fib.ccpHdr. The plcfhdd, a table whose location and length within the file is stored in fib.fcPlcfhdd and fib.cbPlcfhdd, describes where the text of each header/footer begins. If there are **n** headers/footers stored in the Word file, the plcfhdd consists of **n+2** CP entries. The beginning CP of the **ith** header/footer is the **ith** CP in the plcfhdd. The limit CP (the CP of character 1 position past the end of a header/footer) of the **ith** header/footer is the **i+1st** CP in the plcfhdd. **Note**: at the limit CP – 1, Word always places a chEop as a placeholder which is never displayed as part of the header/footer. This allows Word to change an existing header/footer to be empty.

If there are **n** header/footers, the **n+2nd** CP entry value is always 1 greater than the **n+1st** CP entry value. A paragraph end (ASCII 13) is always stored at the file position marked by the **n+1st** CP value.

The transformation in a full saved file from a header/footer CP to an offset from the beginning of a file (fc) is fc=fib.fcMin+ccpText+ccpFtn+cp.

In Word, headers/footers can be defined for a document that will:

1. act as a separator between main text and footnote text.
2. print below footnote text on a page when footnote text must be continued on a succeeding page (continuation separator).
3. print above footnote text on a page when the text must be continued from a previous page (continuation notice).
4. act as a separator between main text and endnote text.
5. print below endnote text on a page when endnote text must be continued on a succeeding page (continuation separator).
6. print above endnote text on a page when the text must be continued from a previous page (continuation notice).

Also for each section defined for the document, distinct headers can be defined for printing on odd-numbered/right facing pages, even-numbered/left facing pages and the first page of a section. Similarly for each document section, distinct footers can be defined for printing on odd-numbered/right facing pages, even-numbered/left facing pages and the first page of a section.

The plcfhdd contains an entry for each kind of header or footer. (The grpfIhdt is no longer used to find entries in the plcfhdd.) Indices in the plcfhdd are as follows:

0 Header for even pages
1 Header for odd pages
2 Footer for even pages
3 Footer for odd pages
4 Header for first page of section
5 Footer for first page of section
6 Footnote separator
7 Footnote continuation separator
8 Footnote continuation notice
9 Endnote separator
10 Endnote continuation separator
11 Endnote continuation notice

Page Table

Page table information is optional data which is not always stored in a Word binary file. It may be stored for the main text, footnote text and endnote text. The fib contains three FCPGD structures (fcpgdMother, fcpgdFtn, fcpgdEdn) which point to where the data is stored. Each fcpgd points to a PLF of PGD structures and a PLCF of BKD structures. The PLF of PGD descriptors contains n entries where n is the number of pages in the associated text stream. The PLC of BKDs contains >= n entries where each entry describes a single break (page break or otherwise) within the text stream. Each BKD is associated with a PGD and contains an ipgd which is an index into the PLF of PGDs. To find the CP range of a given page, traverse the BKDs searching for the first and last BKD which refer to the given page. The CP range of these BKDs is the CP range of the page.

If a Word document is edited in any way, the fcpgds in the fib should be filled with 0s.

Glossary Files

A Word glossary file is a normal Word binary file with two supplemental files, the sttbfglsy, the sttbglsystyle and the plcfglsy, also stored in the file. The sttbfglsy contains a list of the names of glossary entries, the sttbglsystyle contains a list of the style names for every auto text entry, and the plcfglsy contains a table of beginning positions within the text address space of the file of the text of glossary entries.

The sttbfglsy begins with an integer count of bytes of the size of the sttbfglsy (includes the size of the integer count of bytes). If there are **n** glossary entries defined, **n** Pascal-type strings (string preceded by length byte) will follow, concatenated one after the other. Each string storing one glossary entry name. The collection of glossary entry names must be sorted in case-insensitive ascending order (i.e. **a** and **A** are treated as equal). Also the names **date** and **time** must be included in the list of names. The name of the **ith** glossary entry is the **ith** name defined in the sttbfglsy. The extra field in each entry contains an index on the sttbglsystyle that indicates the style name of the first paragraph in plcfglsy.

The sttbglsystyle is not sorted and has no duplicates. Each entry has an extra field indicating how many auto text entries have that style.

If there are **n** glossary entries, the plcfglsy, will consist of **n+2** CP entries. The **ith** CP entry will contain the location of the beginning of the text for the **ith** glossary entry. The **i+1st** CP entry will contain the limit CP of the **ith** glossary entry. The character at a CP position of limit

CP-1 is always a paragraph mark. The **n+2nd** CP entry always contains fib.ccpText+fib.ccpFtn+fib.ccpHdr+1 if there are headers, footers or footnotes stored in the glossary and contains fib.ccpText+fib.ccpFtn+fib.ccpHdr otherwise. The **n+1st** CP entry is always 1 less than the value of the **n+2nd** entry.

The text for the **time** and **date** entries are always a single paragraph mark (ASCII 13).

Routing Slip

A routing slip is stored in the main document stream as an RS (Routing Slip) structure followed by a set of variable length data. After the RS structure are 4 null terminated strings. Each string is preceded by a short integer containing the string length (including the null terminator). The strings are: the subject, the message text, status and title. Following these strings are a variable number (rs.cRecip) of Routing Recipient (RR) records. Each RR record is immediately followed by a variable number (rr.cb) of bytes containing private data, which is in turn followed by a null terminated string containing the recipient name.

Auto Summary

For a document for which AutoSummary View is active (specified in the ASUMYI), the plcfasumy records the result of the last AutoSummary analysis. Each ASUMY in the PLCF gives the AutoSummary level for the text starting at the corresponding CP. The level must be non-negative and no greater than the upper bound specified in the ASUMYI. The ASUMYI specifies the current summary view level. In emphasize view mode, all text at and below the current summary view level is highlighted. In reduce view mode, all text above the current summary view level is hidden.

STTBFASSOC (Table of Associated Strings)

The following are indices into a table of associated strings:

Ibst	Index	Description
ibstAssocFileNext	0	Unused
ibstAssocDot	1	Filename of associated template
ibstAssocTitle	2	Title of document
ibstAssocSubject	3	Subject of document
ibstAssocKeyWords	4	Keywords of document
ibstAssocComments	5	Comments of document
ibstAssocAuthor	6	Author of document
ibstAssocLastRevBy	7	Name of person who last revised the document
ibstAssocDataDoc	8	Filename of data document
ibstAssocHeaderDoc	9	Filename of header document
ibstAssocCriteria1	10	Packed string used by print merge record selection
ibstAssocCriteria2	11	Packed string used by print merge record selection
ibstAssocCriteria3	12	Packed string used by print merge record selection
ibstAssocCriteria4	13	Packed string used by print merge record selection
ibstAssocCriteria5	14	Packed string used by print merge record selection

Ibst	Index	Description
ibstAssocCriteria6	15	Packed string used by print merge record selection
ibstAssocCriteria7	16	Packed string used by print merge record selection
ibstAssocMax	17	Maximum number of strings in string table

The format of the ibstAssocCriteriaX strings are as follows:

```
int   cbIbstAssoc:8;          // BYTE 0  size of ibstAssocCriteriaX string
int   fCompOr:1;              // BYTE 1  set if cond is an or cond
int   iCompOp:7;              // BYTE 1  index of Comparison Operator
char  stMergeField[];         // Name of MergeField
char  stCompInfo[];           // User Supplied Comparison Information
```

Both stMergeField and stCompInfo are variable length character arrays preceded by a length byte.

Structure Definitions

AnnoTation Reference Descriptor for Word 2000 (ATRDPre10)

Word 2000 Annotation Reference Descriptor, now part of a modified Descriptor structure along with newly introduced properties.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	xstUsrInitl	XCHAR[10]			Pascal-style string holding initials of annotation author
20	14	ibst	short			Index into GrpXstAtnOwners
22	16	ak	short	:2	0003	Unused
			short	:14	FFFC	Unused
24	18	grfbmc	uns short			Unused
26	1A	lTagBkmk	long			When not –1, this tag identifies the annotation bookmark that locates the range of CPs in the main document which this annotation references

cbATRDPre10 (count of bytes of ATRDPre10) is 30 (decimal), 1E(hex).

AnnoTation Reference Descriptor Additions for Word 2002 (ATRDPost10)

This structure was introduced in Word 2002 to augment the Annotation Reference Description structure.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	dttm	DTTM			Date and time for the comment
4	4	fResolved	BF	:1	01	Comment has been resolved
			BF	:15	FFFE	Unused

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
6	6	cDepth	int			Depth of the comment
10	A	diatrdParent	Int			Index of the parent of the comment
14	E	fOWSDiscussionItem	BFL	:1	01	Boolean indicating that this is a comment from a web server discussion.
		fInkAtn	BFL	:1	02	Boolean indicating this is an ink annotation.
			BFL	:14	FFFC	Unused

AnnoTation Reference Descriptor (ATRD)

The modified Annotation Reference Descriptor introduced in Word 2002 combining the information from the previous two structures.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	atrdPre10	ATRDre10	30		See structure definition above
20	14	atrdPost10	ATRDPost10	18		See structure definition above

cbATRD (count of bytes of ATRD) is 48 (decimal), 30(hex).

Auto Numbered List Data Descriptor (ANLD)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	anlv	ANLV (see description of ANLV structure)	20		Number only 1 item per table cell
20	14	fNumber1	uns char			Number only 1 item per table cell
21	15	fNumberAcross	uns char			Number across cells in table rows(instead of down)
22	16	fRestartHdn	uns char			Restart heading number on section boundary
23	17	fSpareX	uns char			Unused (should be 0)
24	18	rgxch	array of 32 XCHARs			Characters displayed before/after auto number

cbANLD (count of bytes of ANLD) is 88 (decimal), 58(hex).

Auto Numbered List Data Descriptor for Word 97 (ANLD80)

Same as ANLD but anlv is of type ANLV80 rather than ANLV.

Auto Number Level Descriptor (ANLV)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	nfc	unsigned char			Number format code (use the number format table below)
1	1	cxchTextBefore	unsigned char			Offset into anld.rgxch that is the limit of the text that will be displayed as the prefix of the auto number text
2	2	cxchTextAfter	unsigned char			anld.cxchTextBefore will be the beginning offset of the text in the anld.rgxch that will be displayed as the suffix of an auto number. The sum of anld.cxchTextBefore + anld.cxchTextAfter will be the limit of the auto number suffix in anld.rgxch
3	3	jc	uns char	:2	03	Justification code 0 left justify

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
						1 center 2 right justify 3 left and right justify
		fPrev	uns char	:1	04	When ==1, number generated will include previous levels (used for legal numbering)
		fHang	uns char	:1	08	When ==1, number will be displayed using a hanging indent
		fSetBold	uns char	:1	10	When ==1, boldness of number will be determined by anld.fBold
		fSetItalic	uns char	:1	20	When ==1, italicness of number will be determined by anld.fItalic
		fSetSmallCaps	uns char	:1	40	When ==1, anld.fSmallCaps will determine whether number will be displayed in small caps or not
		fSetCaps	uns char	:1	80	When ==1, anld.fCaps will determine whether number will be displayed capitalized or not
4	4	fSetStrike	uns char	:1	01	When ==1, anld.fStrike will determine whether the number will be displayed using strikethrough or not.
		fSetKul	uns char	:1	02	When ==1, anld.kul will determine the underlining state of the auto number
		fPrevSpace	uns char	:1	04	When ==1, auto number will be displayed with a single prefixing space character
		fBold	uns char	:1	08	Determines boldness of auto number when anld.fSetBold == 1
		FItalic	uns char	:1	10	Determines italicness of auto number when anld.fSetItalic == 1
		fSmallCaps	uns char	:1	20	Determines whether auto number will be displayed using small caps when anld.fSetSmallCaps == 1
		fCaps	uns char	:1	40	Determines whether auto number will be displayed using caps when anld.fSetCaps == 1
		fStrike	uns char	:1	80	Determines whether auto number will be displayed using caps when anld.fSetStrike == 1
5	5	kul	uns char	:3	07	Determines whether auto number will be displayed with underlining when anld.fSetKul == 1
		ico	uns char	:5	F1	Color of auto number for Word 97;

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
						unused in Word 2000
6	6	ftc	short			Font code of auto number
8	8	hps	uns short			Font half point size (or 0=auto)
10	A	iStartAt	uns short			Starting value (0 to 65535)
12	C	dxaIndent				Width of prefix text (same as indent)
14	E	dxaSpace	uns short			Minimum space between number and paragraph
16	F	cv	COLORREF	4		24-bit color for Word 2000

cbANLV (count of bytes of ANLV) is 20 bytes (decimal), 14 bytes (hex).

Number Format Table

nfc value	Numbering scheme
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)
10	Kanji numbering without the digit character (dbnum1).
11	Kanji numbering with the digit character (dbnum2).
12	46 phonetic Katakana characters in "aiueo" order (aiueo).
13	46 phonetic katakana characters in "iroha" order (iroha).
14	Double Byte character
15	Single Byte character
16	Kanji numbering 3 (dbnum3).
17	Kanji numbering 4 (dbnum4).
18	Circle numbering (circlenum).
19	Double-byte Arabic numbering
20	46 phonetic double-byte Katakana characters (*aiueo*dbchar).
21	46 phonetic double-byte katakana characters (*iroha*dbchar).
22	Arabic with leading zero (01, 02, 03, ..., 10, 11)
23	Bullet (no number at all)
24	Korean numbering 2 (ganada).
25	Korean numbering 1 (chosung).
26	Chinese numbering 1 (gb1).
27	Chinese numbering 2 (gb2).
28	Chinese numbering 3 (gb3).
29	Chinese numbering 4 (gb4).
30	Chinese Zodiac numbering 1
31	Chinese Zodiac numbering 2

nfc value	Numbering scheme
32	Chinese Zodiac numbering 3
33	Taiwanese double-byte numbering 1
34	Taiwanese double-byte numbering 2
35	Taiwanese double-byte numbering 3
36	Taiwanese double-byte numbering 4
37	Chinese double-byte numbering 1
38	Chinese double-byte numbering 2
39	Chinese double-byte numbering 3
40	Chinese double-byte numbering 4
41	Korean double-byte numbering 1
42	Korean double-byte numbering 2
43	Korean double-byte numbering 3
44	Korean double-byte numbering 4
45	Hebrew non-standard decimal
46	Arabic Alif Ba Tah
47	Hebrew Biblical standard
48	Arabic Abjad style
49	Hindi vowels
50	Hindi consonants
51	Hindi numbers
52	Hindi descriptive (cardinals)
53	Thai letters
54	Thai numbers
55	Thai descriptive (cardinals
56	Vietnamese descriptive (cardinals)
57	Page Number format - # -
58	Lower case Russian alphabet
59	Upper case Russian alphabet

Auto Number Level Descriptor for Word 97 (ANLV80)

Same as `ANLV` but without the `cv` property.

AutoSummary Analysis (ASUMY)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	`lLevel`	long			AutoSummary level

`cbASUMY` (count of bytes of `ASUMY`) is 4 bytes.

AutoSummary Info (ASUMYI)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	fValid	short	:1	0001	True if the ASUMYI is valid
		fView	short	:1	0002	True if AutoSummary View is active
		iViewBy	short	:2	000C	Display method for AutoSummary View: 0 = Emphasize in current doc 1 = Reduce doc to summary 2 = Insert into doc 3 = Show in new document
		fUpdateProps	short	:1	0010	True if File Properties summary information should be updated after the next summarization
		reserved	short	:11	FFE0	Reserved
2	2	wDlgLevel	short			Dialog summary level
4	4	lHighestLevel	long			Upper bound for lLevel for sentences in this document
8	8	lCurrentLevel	long			Show document sentences at or below this level

cbASUMYI (count of bytes of ASUMYI) is 12 bytes (decimal), C bytes (hex).

Bin Table Entry (BTE)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	pn	long	:22		AutoSummary level
		unused	long	:10		Unused

cbBTE (count of bytes of BTE) is 4 bytes.

BreaK Descriptor (BKD)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	ipgd	short			Except in textbox BKD, index to PGD in plfpgd that describes the page this break is on
0	0	itxbxs	short			In textbox BKD,
2	2	dcpDepend	short			Number of cp's considered for this break; note that the CP's described by cpDepend in this break reside in the next BKD
		icol	uns short	:8	00FF	
		fTableBreak	uns short	:1	0100	When 1, this indicates that this is a table break.
		fColumnBreak	uns short	:1	0200	When 1, this indicates that this is a column break.
		fMarked	uns short	:1	0400	Used temporarily while Word is running.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
		fUnk	uns short	:1	0800	In textbox BKD, when == 1 indicates cpLim of this textbox is not valid
		fTextOverflow	uns short	:1	1000	In textbox BKD, when == 1 indicates that text overflows the end of this textbox

cbBKD (count of bytes of BKD) is 6.

BooKmark First descriptor (BKF)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	ibkl	short			Index to BKL entry in plcfbkl that describes the ending position of this bookmark in the CP stream
2	2	itcFirst	uns short	:7	007F	When bkf.fCol==1, this is the index to the first column of a table column bookmark
		fPub	uns short	:1	0080	When 1, this indicates that this bookmark is marking the range of a Macintosh Publisher section
		itcLim	uns short	:7	7F00	When bkf.fCol==1, this is the index to limit column of a table column bookmark
		fCol	uns short	:1	8000	When 1, this bookmark marks a range of columns in a table specified by (bkf.itcFirst, bkf.itcLim)

cbBKF (count of bytes of BKF) is 4.

BooKmark Lim descriptor (BKL)

The BKL is no longer stored in the plcfbkl or plcfatnbkl, and is instead reconstructed from the plcfbkf or plcfatnbkf when the file is opened.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	ibkf	short			Index to BKF entry in plcfbkf describing the beginning position of this bookmark in the CP stream. If the bkl.ibkf is negative, add on the number of bookmarks recorded in the hplcbkf to the bkl.ibkf to calculate the index to the BKF that corresponds to this entry.

cbBKL (count of bytes of BKL) is 2.

Border Code (BRC)

The BRC is a substructure of the CHP, PAP, PIC, SEP, TAP and TC. See also the BRC structures for older versions.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	cv	COLORREF	4		24-bit border color
4	4	grpfBrc	ulong	4		
4	4	dptLineWidth	long	:8	FF	Width of a single line in 1/8 pt, max of 32 pt.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
		brcType	long	:8	FF00	Border type code: 0 none 1 single 2 thick 3 double 5 hairline 6 dot 7 dash large gap 8 dot dash 9 dot dot dash 10 triple 11 thin-thick small gap 12 thick-thin small gap 13 thin-thick-thin small gap 14 thin-thick medium gap 15 thick-thin medium gap 16 thin-thick-thin medium gap 17 thin-thick large gap 18 thick-thin large gap 19 thin-thick-thin large gap 20 wave 21 double wave 22 dash small gap 23 dash dot stroked 24 emboss 3D 25 engrave 3D codes 64 – 230 represent border art types and are used only for page borders
		dptSpace	long	:5	1F0000	Width of space to maintain between border and text within border Must be 0 when BRC is a substructure of TC. Stored in points.
		fShadow	long	:1	200000	When 1, border is drawn with shadow. Must be 0 when BRC is a substructure of the TC
		fFrame	long	:1	400000	When 1, don't reverse the border.
		unused	long	:9	FF800000	Reserved

cbBRC (count of bytes of BRC) is 8.

Border Code for Word 97 (BRC80)

The BRC80 is a substructure of the CHP, PAP, PIC, SEP, TAP and TC for Word 97**.**

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	dptLineWidth	short	:8	00FF	Width of a single line in 1/8 pt, max of 32 pt.
		brcType	short	:8	FF00	Border type code: 0 none 1 single 2 thick 3 double 5 hairline 6 dot 7 dash large gap 8 dot dash 9 dot dot dash 10 triple 11 thin-thick small gap 12 thick-thin small gap 13 thin-thick-thin small gap 14 thin-thick medium gap 15 thick-thin medium gap 16 thin-thick-thin medium gap 17 thin-thick large gap 18 thick-thin large gap 19 thin-thick-thin large gap 20 wave 21 double wave 22 dash small gap 23 dash dot stroked 24 emboss 3D 25 engrave 3D codes 64 – 230 represent border art types and are used only for page borders
2	2	ico	short	:8	00FF	Color: 0 Auto 1 Black 2 Blue 3 Cyan 4 Green 5 Magenta 6 Red 7 Yellow 8 White 9 DkBlue 10 DkCyan 11 DkGreen 12 DkMagenta 13 DkRed 14 DkYellow 15 DkGray 16 LtGray
		dptSpace	short	:5	1F00	Width of space to maintain between border and text within border Must be 0 when BRC is a substructure of TC. Stored in points.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
		fShadow	short	:1	2000	When 1, border is drawn with shadow Must be 0 when BRC is a substructure of the TC.
		fFrame	short	:1	4000	Don't reverse the border.
			short	:1	8000	Reserved

The size is 4 bytes

Border Code for Windows Word 95 (BRC70)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	dxpLineWidth	Uns short	:3	00FF	Width of a single line: 0-5 thicknesses (3/4, 1 ½, 2 ¼, 3, 4 ½, 6) 6 = dotted 7 = dashed
		brcType	short	:2	FF00	Border type code: 0 none 1 single 2 thick 3 double
		fShadow	short	:1	2000	When 1, border is drawn with shadow
2	2	ico	short	:5	00FF	Color code : 0 Auto 1 Black 2 Blue 3 Cyan 4 Green 5 Magenta 6 Red 7 Yellow 8 White 9 DkBlue 10 DkCyan 11 DkGreen 12 DkMagenta 13 DkRed 14 DkYellow 15 DkGray 16 LtGray
		dxpSpace	short	:5	1F00	Width of space to maintain between border and text within border. Stored in points.

Border Code for WinWord 1.0 (BRC10)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	dxpLine2Width	short	:3	0007	Width of second line of border in pixels
		dxpSpaceBetween	short	:3	0038	Distance to maintain between both lines of border in pixels
		dxpLine1Width	short	:3	01C0	Width of first border line in pixels
		dxpSpace	short	:5	3E00	Width of space to maintain between border and text within border. Must be 0 when BRC is a substructure of the TC.
		fShadow	short	:1	4000	When 1, border is drawn with shadow. Must be 0 when BRC10 is a substructure of the TC.
		fSpare	short	:1	8000	Reserved

The seven types of border lines that WinWord 1.0 supports are coded with different sets of values for dxpLine1Width, dxpSpaceBetween, and dxpLine2 Width.

The border lines and their brc10 settings follow:

Line type	dxpLine1Width	dxpSpaceBetween	dxpLine2Width
no border	0	0	0
single line border	1	0	0
two single line border	1	1	1
fat solid border	4	0	0
thick solid border	2	0	0
dotted border	6 (special value meaning dotted line)	0	0
hairline border	7 (special value meaning hairline)	0	0

When the **no border** settings are stored in the BRC, brc.fShadow and brc.dxpSpace should be set to 0.

cbBRC10 (count of bytes of BRC10) is 2.

Character Properties (CHP)

The CHP is never stored in Word files. It is the result of decompression operations applied to CHPXs. For this reason no offsets are shown into the structure. It can be reconstructed at will.

The CHPX is stored in CHPX FKPS and within the STSH.

Note When a CHPX is stored in an FKP it is prefixed by a one-byte count of bytes that records the size of the non-zero prefix of the CHPX. Since the count of bytes must begin on an even boundary within the FKP followed by the non-zero prefix, it's guaranteed that the int and FC fields of the CHPX are aligned on an odd-byte boundary. The best technique for reconstituting the CHPX is to move the non-zero prefix to the beginning of a local instance of a CHPX that has been cleared to zeros.

Field	Type	Size	Comments
grpfChp	uint	4	Collection of the following 32 flags
fBold	uns long	:1	Text is bold when 1 , and not bold when 0.
fItalic	uns long	:1	Italic when 1, not italic when 0
fRMarkDel	uns long	:1	When 1, text has been deleted and will be displayed with strikethrough when revision marked text is to be displayed
fOutline	uns long	:1	Outlined when 1, not outlined when 0
fFldVanish	uns long	:1	Used internally by Word
fSmallCaps	uns long	:1	Displayed with small caps when 1, no small caps when 0
fCaps	uns long	:1	Displayed with caps when 1, no caps when 0
fVanish	uns long	:1	When 1, text has "hidden" format, and is not displayed unless fPagHidden is set in the DOP
fRMark	uns long	:1	When 1, text is newly typed since the last time revision marks have been accepted and will be displayed with an underline when revision marked text is to be displayed
fSpec	uns long	:1	Character is a Word special character when 1, not a special character when 0
fStrike	uns long	:1	Displayed with strikethrough when 1, no strikethrough when 0
fObj	uns long	:1	Embedded object when 1, not an embedded object when 0
fShadow	uns long	:1	Character is drawn with a shadow when 1; drawn without shadow when 0
fLowerCase	uns long	:1	Character is displayed in lower case when 1. No case transformation is performed when 0. This field may be set to 1 only when chp.fSmallCaps is 1.
fData	uns long	:1	When 1, chp.fcPic points to an FFDATA, the data structure binary data used by Word to describe a form field. The bit chp.fData may only be 1 when chp.fSpec is also 1 and the special character in the document stream that has this property is a chPicture (0x01).
fOle2	uns long	:1	When 1, chp.lTagObj specifies a particular object in the object stream that specifies the particular OLE object in the stream that should be displayed when the chPicture fSpec character that is tagged with the fOle2 is encountered. The bit chp.fOle2 may only be 1 when chp.fSpec is also 1 and the special character in the document stream that has this property is a chPicture (0x01).
fEmboss	uns long	:1	Text is embossed when 1 and not embossed when 0
fImprint	uns long	:1	Text is engraved when 1 and not engraved when 0
fDStrike	uns long	:1	Displayed with double strikethrough when 1, no double strikethrough when 0
fUsePgsuSettings	uns long	:1	Used internally by Word
fBoldBi	uns long	:1	Complex Scripts text is bold when 1

Field	Type	Size	Comments
fComplexScripts	uns long	:1	Complex Scripts text that requires special processing to display and process.
fItalicBi	uns long	:1	Complex Scripts text is italics when 1
fBiDi	uns long	:1	Complex Scripts right-to-left text that requires special processing to display and process (character reordering; contextual shaping; display of combining characters and diacritics; specialized justification rules; cursor positioning).
fIcoBi	uns long	:1	Used internally by Word
fNonGlyph	uns long	:1	Used internally by Word
fBoldOther	uns long	:1	Used internally by Word 97 and earlier versions
fItalicOther	uns long	:1	Used internally by Word 97 and earlier versions
fNoProof	uns long	:1	When set to 1, do not check spelling or grammar
fWebHidden	uns long	:1	Text should be hidden in Web View when set to 1
fFitText	uns long	:1	Fit text when set to 1
fCalc	uns long	:1	Used internally by Word
fFmtLineProp	uns long	:1	Used internally by Word
hps	ushort	2	Font size in half points
ftc	ftc	2	No longer stored
ftcAsci	ftc	2	Font for ASCII text
ftcFE	ftc	2	Font for East Asian text
ftcOther	ftc	2	Font for non-East Asian text
ftcBi	ftc	2	Font for Complex Scripts text
rgftc[iftcCompositeMax]	ftc	8	Array of fonts
dxaSpace	xa	4	Space following each character in the run expressed in twip units.
cv	colorref	4	24-bit color

Field	Type	Size	Comments
ico	short	:5	Color of text for Word 97: 0 Auto 1 Black 2 Blue 3 Cyan 4 Green 5 Magenta 6 Red 7 Yellow 8 White 9 DkBlue 10 DkCyan 11 DkGreen 12 DkMagenta 13 DkRed 14 DkYellow 15 DkGray 16 LtGray
pctCharWidth	ushort	2	Character scale
Lid	lid	2	Language ID (calculated) (see LID table below for values)
rglid[clidChpMax]	lid	6	Array of language IDs: convenient index into the next three properties.
LidDefault	lid	2	Default language ID (see LID table below for values)
LidFE	lid	2	East Asian Language ID (see LID table below for values)
lidBi	lid	2	Complex Scripts language ID (see LID table below for values)
kcd	uns char	:3	Emphasis mark: 0 None 1 Dot 2 Comma 3 Circle 4 Under Dot
fUndetermine	uns char	:1	Character is undetermined when set to 1
iss	uns char	:3	Superscript/subscript indices 0 means no super/subscripting 1 means text in run is superscripted 2 means text in run is subscripted
fSpecSymbol	uns char	:1	Used by Word internally
idct	uchar	1	Not stored in file
idctHint	uchar	1	Identifier of Character type

Field	Type	Size	Comments
kul	uchar	1	Underline code: 0 none 1 single 2 by word 3 double 4 dotted 5 hidden 6 thick 7 dash 8 dot (not used) 9 dot dash 10 dot dot dash 11 wave 20 kulDottedHeavy 23 kulDashedHeavy 25 kulDotDashHeavy 26 kul2DotDashHeavy 27 kulWaveHeavy 39 kulDashLong 43 kulWaveDouble 55 kulDashLongHeavy
hres	uchar	1	Hyphenation rule 0 No hyphenation 1 Normal hyphenation 2 Add letter before hyphen 3 Change letter before hyphen 4 Delete letter before hyphen 5 Change letter after hyphen 6 Delete letter before the hyphen and change the letter preceding the deleted character
chHres	uchar	1	The character that will be used to add or change a letter when chp.ysr is 2,3, 5 or 6
hpsKern	ushort	2	Kerning distance for characters in run recorded in half points
hpsPos	short	2	Reserved
cvUl	colorref	4	Underline color
shd	shd	10	Shading
brc	brc	8	Border
ibstRMark	ibst	2	Index to author IDs stored in hsttbfRMark. Used when text in run was newly typed when revision marking was enabled.
sfxtText	uchar	1	Text animation: 0 no animation 1 Las Vegas lights 2 background blink 3 sparkle text 4 marching ants 5 marching red ants 6 shimmer
fDblBdr	uns char	:1	Used internally by Word
fBorderWS	uns char	:1	Used internally by Word

Field	Type	Size	Comments
ufel	ushort	2	Collection properties represented by itypFELayout and copt (East Asian layout properties)
itypFELayout	uchar	1	Collection of the next 6 properties: 0x00 – none 0x01 – Tatenakayoko 0x02 – Warichu 0x04 – Kumimoji 0xFF – All
fTNY	uns char	:1	Tatenakayoko: Horizontal–in-vertical (range of text in a direction perpendicular to the text flow) is used when set to 1
fWarichu	uns char	:1	Two lines in one (text in the group is displayed as two half-height lines within a line) when set to 1
fKumimoji	uns char	:1	When set to 1, combine characters
fRuby	uns char	:1	Phonetic guide when set to 1
fLSFitText	uns char	:1	When set to 1, fit text
spare	Uns char	:3	Unused
copt	uchar	1	Collection of the following 5 flags
iWarichuBracket	uns char	:3	Bracket character for two-lines-in-one: 0 text is not enclosed. 1 text is enclosed in parentheses. 2 text is enclosed in square brackets ([]). 3 text is enclosed in angled brackets (<>). 4 text is enclosed in braces ({}).
fWarichuNoOpenBracket	uns char	:1	Two-lines-in-one uses no open bracket when 1
fTNYCompress	uns char	:1	When 1, fit text in line
fTNYFetchTxm	uns char	:1	When 1, fetch text metrics
fCellFitText	uns char	:1	When 1, Fit text in cell
unused	uns char	:1	Not used
hpsAsci	ushort	2	Font size for ASCII font
hpsFE	ushort	2	Font size for East Asian text
hpsBi	ushort	2	Font size for Complex Scripts text
rghps[ihpsMax]	ushort	6	Array of font sizes
ftcSym	ftc	2	When chp.fSpec is 1 and the character recorded for the run in the document stream is chSymbol (0x28), chp.ftcSym identifies the font code of the symbol font that will be used to display the symbol character recorded in chp.xchSym chp.ftcSym is an index into the rgffn structure.
xchSym	xchar	2	When chp.fSpec==1 and the character recorded for the run in the document stream is chSymbol (0x28), the character stored chp.xchSym will be displayed using the font specified in chp.ftcSym

Field	Type	Size	Comments
hpsAsci	ushort	2	Font size for ASCI font
hpsFE	ushort	2	Font size for East Asian text
hpsBi	ushort	2	Font size for Complex Scripts text
rghps[ihpsMax]	ushort	6	Array of font sizes
fNumRunBi	uns short	:1	Used internally by Word
fSysVanish	uns short	:1	Used internally by Word
fDiacRunBi	uns short	:1	Used internally by Word
fBoldPresent	uns short	:1	Used internally by Word
fItalicPresent	uns short	:1	Used internally by Word
fcPic	fc	4	Offset in data stream pointing to beginning of a picture when character is a picture character (character is 0x01 and chp.fSpec is 1).
fcObj	fc	4	Offset in data stream pointing to beginning of a picture when character is an OLE1 object character (character is 0x20 and chp.fSpec is 1, chp.fOle2 is 0).
lTagObj	ulong	4	An object ID for an OLE object, only set if chp.fSpec and chp.fOle2 are both true, and chp.fObj.
fcData	fc	4	Points to location of picture data, only if chp.fSpec is true.
hsp	msohsp	4	Used internally by Word
docPic	ulong	:16	Used internally by Word
dummy	ulong	:15	Used internally by Word
fDirty	ulong	:1	Used internally by Word
hresOld	ulong	:8	Hyphenation rule 0 No hyphenation 1 Normal hyphenation 2 Add letter before hyphen 3 Change letter before hyphen 4 Delete letter before hyphen 5 Change letter after hyphen 6 Delete letter before the hyphen and change the letter preceding the deleted character
chHresOld	ulong	:8	The character that will be used to add or change a letter when chp.hresi is 2,3, 5 or 6
dummy3	ulong	:16	Not used
dxpKashida	xa	4	Used internally by Word
dxpSpace	xa	4	Used internally by Word
ibstRMarkDel	ibst	2	Index to author IDs stored in hsttbfRMark. Used when text in run was deleted when revision marking was enabled.

Field	Type	Size	Comments
dttmRMark	dttm	4	Date/time at which this run of text was entered/modified by the author (Only recorded when revision marking is on.)
dttmRMarkDel	dttm	4	Date/time at which this run of text was deleted by the author (Only recorded when revision marking is on.)
cHpsInc	short	2	Used internally by Word
istd	ushort	2	Index to character style descriptor in the stylesheet that tags this run of text. When istd is istdNormalChar (10 decimal), characters in run are not affected by a character style. If chp.istd contains any other value, chpx of the specified character style are applied to CHP for this run before any other exceptional properties are applied.
idslRMReason	ushort	2	An index to strings displayed as reasons for actions taken by Word's AutoFormat code
idslRMReasonDel	ushort	2	An index to strings displayed as reasons for actions taken by Word's AutoFormat code
cpg	ushort	2	Code page of run in pre-Unicode files
iatrUndetType	uns short	:4	Used internally by Word
fUlGap	uns short	:1	Used internally by Word 8
icoHighlight	uns short	:5	Highlight color (see chp.ico)
fHighlight	uns short	:1	When 1, characters are highlighted with color specified by chp.icoHighlight.
fScriptAnchor	uns short	:1	Used internally by Word
fFixedObj	uns short	:1	Used internally by Word
spare2	uns short	:1	Not used
fNavHighlight	uns short	:1	Used internally by Word
fChsDiff	uns short	:1	Pre-Unicode files, char's char set different from FIB char set
fMacChs	uns short	:1	fTrue if char's are Macintosh char set
fFtcAsciSym	uns short	:1	Used internally by Word
fFtcReq	uns short	:1	Used internally by Word
fLangApplied	uns short	:1	Used internally by Word
fSpareLangApplied	uns short	:1	Used internally by Word
fForcedCvAuto	uns short	:1	Used internally by Word
fPropRMark	uns short	2	When 1, properties have been changed with revision marking on
ibstPropRMark	ibst	2	Index to author IDs stored in hsttbfRMark. Used when properties have been changed when revision marking was enabled.
dttmPropRMark	dttm	4	Date/time at which properties of this were changed for this run of text by the author. (Only recorded when revision marking is on.)
fAnmPropRMark	byte	1	Used internally by Word

Field	Type	Size	Comments
fConflictOrig	uchar	1	When chp.wConflict!=0, this is fTrue when text is part of the original version of text. When fFalse, text is alternative introduced by reconciliation operation.
fConflictOtherDel	uchar	1	When fConflictOtherDel==fTrue, the other side of a reconciliation conflict causes this text to be deleted.
wConflict	ushort	2	When != 0, index number that identifies all text participating in a particular conflict incident.
IbstConflict	ibst	2	Who made this change for this side of the conflict.
dttmConflict	dttm	4	When the change was made.
fDispFldRMark	byte	1	When 1, the number for a ListNum field is being tracked in xstDispFldRMark. If that number is different from the current value, the number has changed. Only valid for ListNum fields.
ibstDispFldRMark	ibst	2	Index to author IDs stored in hsttbfRMark. Used when ListNum field numbering has been changed when revision marking was enabled.
dttmDispFldRMark	dttm	4	The date for the ListNum field number change.
xstDispFldRMark	xstdispfld	32	The string value of the ListNum field when revision mark tracking began.
dxtSpaceExtra	x	4	Used internally by Word.
fcObjp	fc	4	Offset in the data stream indicating the location of OLE object data.
dxaFitText	xa	4	Fit Text Width.
lFitTextID	long	4	Serialize Fit Text Areas.
lbrCRJ	uchar	1	Line BReak code for xchCRJ: 0 lbrNone 1 lbrLeft 2 lbrRight 3 lbrBoth
iuhi	long	4	Unknown HTML element
bTransNoProof0	uchar	1	Used internally to handle translating Word 97 lid sprms into no proofing and Word 2000 sprms.
bTransNoProof1	uchar	1	Ued internally to handle translating Word 97 lid sprms into no proofing and Word 2000 sprms.
rsidProp	RSID	4	Save ID for last time this CHP was revised: a random number associated with character formatting which improves the accuracy of Word's document merge feature.
rsidText	RSID	4	Save ID for last time this text was revised: a random number associated with the insertion of text which improves the accuracy of Word's document merging.
rsidRMDel	RSID	4	Save ID for last time this revision-mark-deleted text was revised: a random number associated with the tracked deletion of text which improves the accuracy of Word's document merging.

Field	Type	Size	Comments
fSpecVanish	uchar	:1	Special hidden for leading emphasis (always hidden).
fHasOldProps	uchar	1	Used for character property revision marking. The chp at the time fHasOldProps is set to 1, the is the old chp.
pbi	PBI	6	Picture bullet information
hplcnf	HPL	4	Conditional character formatting for table styles. No language properties are stored here.
ffm	uchar	1	font fixup mode, for internal use only.
fSdtVanish	uchar	1	Mark the character as hidden.

The standard CHP is all zeros except:

hps	20 half-points
fcPic	-1
istd	10 (the standard character style)
lidDefault, lidFE	0x0400 (no proofing)
wCharScale	100
fUsePgsuSettings	-1

Table of LIDs

Language	ID (decimal)	ID (hex)
(no proofing)	1024	400
Afrikaans	1078	436
Albanian	1052	41C
Amharic	1118	45E
Arabic (Algeria)	5121	1401
Arabic (Bahrain)	15361	3C01
Arabic (Egypt)	3073	C01
Arabic (Iraq)	2049	801
Arabic (Jordan)	11265	2C01
Arabic (Kuwait)	13313	3401
Arabic (Lebanon)	12289	3001
Arabic (Libya)	4097	1001
Arabic (Morocco)	6145	1801
Arabic (Oman)	8193	2001
Arabic (Qatar)	16385	4001
Arabic (Saudi Arabia)	1025	401
Arabic (Syria)	10241	2801
Arabic (Tunisia)	7169	1C01
Arabic (U.A.E)	14337	3801
Arabic (Yemen)	9217	2401
Armenian	1067	42B
Assamese	1101	44D

Language	ID (decimal)	ID (hex)
Azeri (Cyrillic)	2092	82C
Azeri (Latin)	1068	42C
Basque	1069	42D
Belarusian	1059	423
Bengali	1093	445
Bengali (Bangladesh)	2117	845
Bulgarian	1026	402
Burmese	1109	455
Catalan	1027	403
Cherokee	1116	45C
Chinese (Hong Kong S.A.R.)	3076	C04
Chinese (Macao S.A.R.)	5124	1404
Chinese (PRC)	2052	804
Chinese (Singapore)	4100	1004
Chinese (Taiwan)	1028	404
Croatian	1050	41A
Czech	1029	405
Danish	1030	406
Divehi	1125	465
Dutch (Belgium)	2067	813
Dutch (Netherlands)	1043	413
Edo	1126	466
English (Australia)	3081	C09
English (Belize)	10249	2809
English (Canada)	4105	1009
English (Caribbean)	9225	2409
English (Hong Kong S.A.R.)	15369	3C09
English (India)	16393	4009
English (Indonesia)	14345	3809
English (Ireland)	6153	1809
English (Jamaica)	8201	2009
English (Malaysia)	17417	4409
English (New Zealand)	5129	1409
English (Philippines)	13321	3409
English (Singapore)	18441	4809
English (South Africa)	7177	1C09
English (Trinidad and Tobago)	11273	2C09
English (U.K.)	2057	809
English (U.S.)	1033	409
English (Zimbabwe)	12297	3009
Estonian	1061	425
Faeroese	1080	438
Farsi	1065	429
Filipino	1124	464

Language	ID (decimal)	ID (hex)
Finnish	1035	40B
French (Belgium)	2060	80C
French (Cameroon)	11276	2C0C
French (Canada)	3084	C0C
French (Congo (DRC))	9228	240C
French (Cote d'Ivoire)	12300	300C
French (France)	1036	40C
French (Haiti)	15372	3C0C
French (Luxembourg)	5132	140C
French (Mali)	13324	340C
French (Monaco)	6156	180C
French (Morocco)	14348	380C
French (Reunion)	8204	200C
French (Senegal)	10252	280C
French (Switzerland)	4108	100C
French (West Indies)	7180	1C0C
Frisian (Netherlands)	1122	462
Fulfulde	1127	467
FYRO Macedonian	1071	42F
Gaelic (Ireland)	2108	83C
Gaelic (Scotland)	1084	43C
Galician	1110	456
Georgian	1079	437
German (Austria)	3079	C07
German (Germany)	1031	407
German (Liechtenstein)	5127	1407
German (Luxembourg)	4103	1007
German (Switzerland)	2055	807
Greek	1032	408
Guarani	1140	474
Gujarati	1095	447
Hausa	1128	468
Hawaiian	1141	475
Hebrew	1037	40D
Hindi	1081	439
Hungarian	1038	40E
Ibibio	1129	469
Icelandic	1039	40F
Igbo	1136	470
Indonesian	1057	421
Inuktitut	1117	45D
Italian (Italy)	1040	410
Italian (Switzerland)	2064	810
Japanese	1041	411

Language	ID (decimal)	ID (hex)
Kannada	1099	44B
Kanuri	1137	471
Kashmiri	2144	860
Kashmiri (Arabic)	1120	460
Kazakh	1087	43F
Khmer	1107	453
Konkani	1111	457
Korean	1042	412
Kyrgyz	1088	440
Lao	1108	454
Latin	1142	476
Latvian	1062	426
Lithuanian	1063	427
Malay	1086	43E
Malay (Brunei Darussalam)	2110	83E
Malayalam	1100	44C
Maltese	1082	43A
Manipuri	1112	458
Maori	1153	481
Marathi	1102	44E
Mongolian	1104	450
Mongolian (Mongolian)	2128	850
Nepali	1121	461
Nepali (India)	2145	861
Norwegian (Bokmål)	1044	414
Norwegian (Nynorsk)	2068	814
Oriya	1096	448
Oromo	1138	472
Papiamentu	1145	479
Pashto	1123	463
Polish	1045	415
Portuguese (Brazil)	1046	416
Portuguese (Portugal)	2070	816
Punjabi	1094	446
Punjabi (Pakistan)	2118	846
Quechua (Bolivia)	1131	46B
Quechua (Ecuador)	2155	86B
Quechua (Peru)	3179	C6B
Rhaeto-Romanic	1047	417
Romanian (Moldova)	2072	818
Romanian (Romania)	1048	418
Russian (Moldova)	2073	819
Russian (Russia)	1049	419
Sami (Lappish)	1083	43B

Language	ID (decimal)	ID (hex)
Sanskrit	1103	44F
Sepedi	1132	46C
Serbian (Cyrillic)	3098	C1A
Serbian (Latin)	2074	81A
Sindhi (Arabic)	2137	859
Sindhi (Devanagari)	1113	459
Sinhalese	1115	45B
Slovak	1051	41B
Slovenian	1060	424
Somali	1143	477
Sorbian	1070	42E
Spanish (Argentina)	11274	2C0A
Spanish (Bolivia)	16394	400A
Spanish (Chile)	13322	340A
Spanish (Colombia)	9226	240A
Spanish (Costa Rica)	5130	140A
Spanish (Dominican Republic)	7178	1C0A
Spanish (Ecuador)	12298	300A
Spanish (El Salvador)	17418	440A
Spanish (Guatemala)	4106	100A
Spanish (Honduras)	18442	480A
Spanish (Mexico)	2058	80A
Spanish (Nicaragua)	19466	4C0A
Spanish (Panama)	6154	180A
Spanish (Paraguay)	15370	3C0A
Spanish (Peru)	10250	280A
Spanish (Puerto Rico)	20490	500A
Spanish (Spain-Modern Sort)	3082	C0A
Spanish (Spain-Traditional Sort)	1034	40A
Spanish (Uruguay)	14346	380A
Spanish (Venezuela)	8202	200A
Sutu	1072	430
Swahili	1089	441
Swedish (Finland)	2077	81D
Swedish (Sweden)	1053	41D
Syriac	1114	45A
Tajik	1064	428
Tamazight	1119	45F
Tamazight (Latin)	2143	85F
Tamil	1097	449
Tatar	1092	444
Telugu	1098	44A
Thai	1054	41E
Tibetan (Bhutan)	2129	851

Language	ID (decimal)	ID (hex)
Tibetan (PRC)	1105	451
Tigrigna (Eritrea)	2163	873
Tigrigna (Ethiopia)	1139	473
Tsonga	1073	431
Tswana	1074	432
Turkish	1055	41F
Turkmen	1090	442
Ukrainian	1058	422
Urdu	1056	420
Uzbek (Cyrillic)	2115	843
Uzbek (Latin)	1091	443
Venda	1075	433
Vietnamese	1066	42A
Welsh	1106	452
Xhosa	1076	434
Yi	1144	478
Yiddish	1085	43D
Yoruba	1130	46A
Zulu	1077	435

Character Property Exceptions (CHPX)

The CHPX is stored within **Character** FKPs and within the STSH in STDs for **paragraph style** and **character style** entries.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	cb	byte			Count of bytes of following data in CHPX
1	1	grpprl	character array			A list of the sprms that encode the differences between CHP for a run of text and the CHP generated by the paragraph and character styles that tag the run.

Date and Time (internal date format) (DTTM)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
0	0	mint	short	:6	003F	minutes (0-59)
		hr	short	:5	07C0	hours (0-23)
		dom	short	:5	F800	days of month (1-31)
2	2	mon	short	:4	000F	months (1-12)
		yr	short	:9	1FF0	years (1900-2411)-1900
		wdy	short	:3	E000	weekday Sunday=0 Monday=1 Tuesday=2 Wednesday=3 Thursday=4 Friday=5 Saturday=6

cbDTTM (count of bytes of DTTM) is 4.

Drop Cap Specifier(DCS)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Default value	Comment
0	0	fdct	short	:3	0007	0	Drop cap type 0 no drop cap 1 normal drop cap 2 drop cap in margin
			short	:5	00F8	0	Count of lines to drop
1	1		short	:8			Reserved

cbDCS (count of bytes of DCS) is 2.

Document Properties (DOP)

Each version of Word, the DOP gets a little bit larger. Shown below are four different versions of the DOP: for nFib values < 103, for nFib values between 103 and 105, for nFib values greater than 105 and less than 217, and for nFib values greater than or equal to 217. Word 97 and later versions write files with nFib>105. Word 6.0 for the Macintosh writes files with nFib==103 or nFib==104. The compatibility options (copts) section was grown (to add more compatibility options in the Tools/Options/Compatibility dialog) and copied to the end of the DOP, so for files with nFib>=103, the first copts section should be ignored (and the analogous fields in the new copts section used instead), whereas files with nFib<103 will have DOP's without the new copts section. See below for the addition.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Default value	comment
0	0	fFacingPages	short	:1	0001	0	1 when facing pages should be printed
		fWidowControl	short	:1	0002	1	1 when widow control is in effect; 0 when widow control disabled
		fPMHMainDoc	short	:1	0004	0	1 when doc is a main doc for Print Merge Helper, 0 when not; default=0
		grfSuppression	short	:2	0018	0	Default line suppression storage; 0= form letter line suppression; 1= no line suppression; default=0. No longer used.
		Fpc	short	:2	0060	1	Footnote position code 0 print as endnotes 1 print at bottom of page 2 print immediately beneath text
			short	:1	0080	0	Unused
1	1	grpfIhdt	short	:8	FF00	0	No longer used.
2	2	rncFtn	short	:2	0003	0	Restart index for footnotes 0 don't restart note numbering 1 restart for each section 2 restart for each page
		nFtn	short	:14	FFFC	1	Initial footnote number for document
4	4	fOutlineDirtySave	short	:1	0001		When 1, indicates that information in the hplcpad should be refreshed since outline has been dirtied
			short	:7	00FE		Reserved
5	5	fOnlyMacPics	short	:1	0100		When 1, Word believes all pictures recorded in the document were created on a Macintosh
		fOnlyWinPics	short	:1	0200		When 1, Word believes all pictures recorded in the document were created in Windows
		fLabelDoc	short	:1	0400		When 1, document was created as a print merge labels document

b_{10}	b_{16}	Field	Type	Size	Bitfield	Default value	comment
		fHyphCapitals	short	:1	0800		When 1, Word is allowed to hyphenate words that are capitalized. When 0, capitalized may not be hyphenated
		fAutoHyphen	short	:1	1000		When 1, Word will hyphenate newly typed text as a background task
		fFormNoFields	short	:1	2000		
		fLinkStyles	short	:1	4000		When 1, Word will merge styles from its template
		fRevMarking	short	:1	8000		When 1, Word will mark revisions as the document is edited
6	6	fBackup	short	:1	0001		When 1, always make backup when document saved
		fExactCWords	short	:1	0002		When 1, the results of the last Word Count execution (as recorded in several DOP fields) are still exactly correct
		fPagHidden	short	:1	0004		When 1, hidden document contents are displayed
		fPagResults	short	:1	0008		When 1, field results are displayed, when 0 field codes are displayed
		fLockAtn	short	:1	0010		When 1, annotations are locked for editing
		fMirrorMargins	short	:1	0020		When 1, swap margins on left/right pages
			short	:1			Reserved
		fDfltTrueType	short	:1	0080		When 1, use TrueType fonts by default (flag obeyed only when doc was created by WinWord 2.x)
7	7	fPagSuppressTopSpacing	short	:1	0100		When 1, file created with SUPPRESSTOPSPACING=YES in WIN.INI. (flag obeyed only when doc was created by WinWord 2.x).
		fProtEnabled	short	:1	0200		When 1, document is protected from edit operations
		fDispFormFldSel	short	:1	0400		When 1, restrict selections to occur only within form fields
		fRMView	short	:1	0800		When 1, show revision markings on screen
		fRMPrint	short	:1	1000		When 1, print revision marks when document is printed
			short	:1			Reserved

b_{10}	b_{16}	Field	Type	Size	Bitfield	Default value	comment
		fLockRev	short	:1	4000		When 1, the current revision marking state is locked
		fEmbedFonts	short	:1	8000		When 1, document contains embedded TrueType fonts
8	8	fNoTabForInd	short	:1	0001		Compatibility option: when 1, don't add automatic tab stops for hanging indent
		fNoSpaceRaiseLower		:1	0002		Compatibility option: when 1, don't add extra space for raised or lowered characters
		fSuppressSpbfAfterPageBreak:		:1	0004		Compatibility option: when 1, suppress the paragraph Space Before and Space After options after a page break
		fWrapTrailSpaces		:1	0008		Compatibility option: when 1, wrap trailing spaces at the end of a line to the next line
		fMapPrintTextColor		:1	0010		Compatibility option: when 1, print colors as black on non-color printers
		fNoColumnBalance		:1	0020		Compatibility option: when 1, don't balance columns for Continuous Section starts
		fConvMailMergeEsc		:1	0040		
		fSupressTopSpacing		:1	0080		Compatibility option: when 1, suppress extra line spacing at top of page
		fOrigWordTableRules		:1	0100		Compatibility option: when 1, combine table borders like Word 5.x for the Macintosh
		fTransparentMetafiles		:1	0200		Compatibility option: when 1, don't blank area between metafile pictures
		fShowBreaksInFrames		:1	0400		Compatibility option: when 1, show hard page or column breaks in frames
		fSwapBordersFacingPgs		:1	0800		Compatibility option: when 1, swap left and right pages on odd facing pages
					F000		Reserved
10	A	dxaTab	uns short			720 twips	Default tab width
12	C	wSpare	uns short				Reserved
14	E	dxaHotZ	uns short				Width of hyphenation hot zone measured in twips
16	10	cConsecHypLim	uns short				Number of lines allowed to have consecutive hyphens

b_{10}	b_{16}	Field	Type	Size	Bitfield	Default value	comment
18	12	wSpare2	uns short				Reserved
20	14	dttmCreated	DTTM				Date and time document was created
24	18	dttmRevised	DTTM				Date and time document was last revised
28	1C	dttmLastPrint	DTTM				Date and time document was last printed
32	20	nRevision	int				Number of times document has been revised since its creation
34	22	tmEdited	long				Time document was last edited
38	26	cWords	long				Count of words tallied by last Word Count execution
42	2A	cCh	long				Count of characters tallied by last Word Count execution
46	2E	cPg	int				Count of pages tallied by last Word Count execution
48	30	cParas	long				Count of paragraphs tallied by last Word Count execution
52	34	rncEdn	short	:2	0003		Restart endnote number code 0 don't restart endnote numbering 1 restart for each section 2 restart for each page
		nEdn	short	:14	FFFC		Beginning endnote number
54	36	Epc	short	:2	0003		Endnote position code 0 display endnotes at end of section 3 display endnotes at end of document
		nfcFtnRef	short	:4	003C		Number format code for auto footnotes. Use the Number Format Table below. Note: only the first 16 values in the table can be used.
		nfcEdnRef	short	:4	03C0		Number format code for auto endnotes. Use the Number Format Table below. Note: only the first 16 values in the table can be used.
		fPrintFormData	short	:1	0400		Only print data inside of form fields
		fSaveFormData	short	:1	0800		Only save document data that is inside of a form field
		fShadeFormData	short	:1	1000		Shade form fields
				:2	6000		Reserved

b_{10}	b_{16}	Field	Type	Size	Bitfield	Default value	comment
		fWCFtnEdn	short	:1	8000		When 1, include footnotes and endnotes in word count
56	38	cLines	long				Count of lines tallied by last Word Count operation
60	3C	cWordsFtnEnd	long				Count of words in footnotes and endnotes tallied by last word count operation
64	40	cChFtnEdn	long				Count of characters in footnotes and endnotes tallied by last word count operation
68	44	cPgFtnEdn	short				Count of pages in footnotes and endnotes tallied by last word count operation
70	46	cParasFtnEdn	long				Count of paragraphs in footnotes and endnotes tallied by last word count operation
74	4A	cLinesFtnEdn	long				Count of paragraphs in footnotes and endnotes tallied by last word count operation
78	4E	lKeyProtDoc	long				Document protection password key, only valid if dop.fProtEnabled, dop.fLockAtn or dop.fLockRev is 1
82	52	wvkSaved	short	:3	0007		Document view kind 0 Normal view 1 Outline view 2 Page View
		wScaleSaved	short	:9	0FF8		Zoom percentage
		zkSaved	short	:2	3000		Zoom type 0 None 1 Full page 2 Page width
		fRotateFontW6	short	:1	4000		This is a vertical document (Word 95 and Word 6.0 for Windows only)
		iGutterPos	short	:1	8000		Gutter position for this doc: 0 => side; 1 => top

In a file with nFib < 103—for example, documents created with Word 6.0 for Windows—the DOP would end here. This DOP would have a cbDOP of 84, and a cwDOP of 42.

Files with nFib >= 103, the compatibility options (copts) section at offset 8 was copied here and expanded. Options marked "(see above)" hold the same value that the same-named field in the old copts section above had in files with nFib < 103.

b10	b16	Field	Type	Size	Bitfield	Comment
84	54	fNoTabForInd	uns long	:1	00000001	(see above)
		fNoSpaceRaiseLower		:1	00000002	(see above)
		fSupressSpbfAfterPageBreak		:1	00000004	(see above)
		fWrapTrailSpaces		:1	00000008	(see above)
		fMapPrintTextColor		:1	00000010	(see above)
		fNoColumnBalance		:1	00000020	(see above)
		fConvMailMergeEsc		:1	00000040	(see above)
		fSupressTopSpacing		:1	00000080	(see above)
		fOrigWordTableRules		:1	00000100	(see above)
		fTransparentMetafiles		:1	00000200	(see above)
		fShowBreaksInFrames		:1	00000400	(see above)
		fSwapBordersFacingPgs		:1	00000800	(see above)
				:4	0000F000	(reserved)
		fSuppressTopSpacingMac5		:1	00010000	Suppress extra line spacing at top of page like Word 5.x for the Macintosh
		fTruncDxaExpand		:1	00020000	Expand/Condense by whole number of points
		fPrintBodyBeforeHdr		:1	00040000	Print body text before header/footer
		fNoLeading		:1	00080000	Don't add leading (extra space) between rows of text
				:1	00100000	(reserved)
		fMWSmallCaps		:1	00200000	Use larger small caps like Word 5.x for the Macintosh
				:10	FFC00000	(reserved)

For this expanded DOP, cbDOP=88 and cwDOP=44.

For files with nFib=106 (Word 97), the DOP has a number of additional fields:

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
88	58	adt	short			Autoformat document type: 0 for normal. 1 for letter, and 2 for email.
90	5A	doptypography	DOPTYPOGRAPHY			See DOPTYPOGRAPHY
400	190	dogrid	DOGRID			See DOGRID
410	19A	reserved	short	:1	0001	Always set to zero when writing files

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
		lvl	short	:4	001E	Which outline levels are showing in outline view (0 => heading 1 only, 4 => headings 1 through 5, 9 => all levels showing)
		fGramAllDone	short	:1	0020	Document has been completely grammar checked
		fGramAllClean	short	:1	0040	No grammar errors exist in document
		fSubsetFonts	short	:1	0080	If you are doing font embedding, you should only embed the characters in the font that are used in the document
		fHideLastVersion	short	:1	0100	Hide the version created for auto version
		fHtmlDoc	short	:1	0200	This file is based upon an HTML file
		reserved	short	:1	0400	Always set to zero when writing files
		fSnapBorder	short	:1	0800	Snap table and page borders to page border
		fIncludeHeader	short	:1	1000	Place header inside page border
		fIncludeFooter	short	:1	2000	Place footer inside page border
		fForcePageSizePag	short	:1	4000	Are we in online view
		fMinFontSizePag	short	:1	8000	Are we auto-promoting fonts to >= hpsZoonFontPag?
412	19C	fHaveVersions	short	:1	0001	Versioning is turned on
		fAutoVersion	short	:1	0002	Auto versioning is enabled
		reserved	short	:14	FFFC	Always set to zero when writing files
414	19E	asumyi	ASUMYI			Auto summary info
426	1AA	cChWS	long			Count of characters with spaces
430	1AE	cChWSFtnEdn	long			Count of characters with spaces in footnotes and endnotes
434	1B2	grfDocEvents	long			
438	1B6	fVirusPrompted	long	:1	0001	Have we prompted for virus protection on this document?
		fVirusLoadSafe	long	:1	0002	If prompted, load safely for this document?
		KeyVirusSession30	long	:30	FFFC	Random session key to sign above bits for a Word session
442	1BA	Spare	30 bytes			Spare

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
472	1D8	reserved	long			Always set to zero when writing files
476	1DC	reserved	long			Always set to zero when writing files
480	1E0	cDBC	long			Count of double byte characters
484	1E4	cDBCFtnEdn	long			Count of double byte characters in footnotes and endnotes
488	1E8	reserved	long			Always set to zero when writing files
492	1EC	nfcFtnRef	short			Number format code for auto footnote references (use the Number Format Table below)
494	1EE	nfcEdnRef	short			Number format code for auto endnote references (use the Number Format Table below)
496	1F0	hpsZoonFontPag	short			Minimum font size if fMinFontSizePag is true
498	1F2	dywDispPag	short			Height of the window in online view during last repagination

For this expanded DOP, cbDOP=88 and cwDOP=44.

For files with nFib>217 (Word 2000, 2002, and 2003) the following was added:

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
500	1F4	ilvlLastBulletMain	uchar	1		Used internally by Word
501	1F5	ilvlLastNumberMain	uchar	1		Used internally by Word
502	1F6	istdClickTypePara	ushort	2		Default paragraph style for click and type
504	1F8	fLADAllDone	uns short	:1	00000001	When set to 1, language of all text in doc has been auto-detected
504	1F8	fEnvelopeVis	uns short	:1	00000002	When set to 1, envelope is visible.
504	1F8	fMaybeTentativeListInDoc	uns short	:1	00000004	When set to 1, doc may have a tentative list in it
504	1F8	fMaybeFitText	uns short	:1	00000006	When set to 1, doc may have fit text
504	1F8	Empty	uns short	:5	000001F2	Web options begin.
504	1F8	fRelyOnCSS_WebOpt	uns short	:1	00000200	When set to 1, rely on CSS for formatting

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
504	1F8	fRelyOnVML_WebOpt	uns short	:1	00000400	When set to 1, Rely on VML for displaying graphics in browsers
504	1F8	fAllowPNG_WebOpt	uns short	:1	00000800	When set to 1, allow PNG as an output format for graphics
504	1F8	screenSize_WebOpt	uns short	:4	0000F000	Target monitor screen size
506	1FA	fOrganizeInFolder_WebOpt	uns short	:1	00000001	When set to 1, organize supporting files in a folder
506	1FA	fUseLongFileNames_WebOpt	uns short	:1	00000002	Use long file names for supporting files
506	1FA	iPixelsPerInch_WebOpt	uns short	:10	00000FFC	Target monitor resolution in pixels per inch
506	1FA	fWebOptionsInit	uns short	:1	00001000	When set to 1, the web options have been filled in
506	1FA	fMaybeFEL	uns short	:1	00002000	When set to 1, the document may have East Asian layouts
506	1FA	fCharLineUnits	uns short	:1	00004000	When set to 1, there may be character unit indents or line unit
506	1FA	fMaybeRTLTables	uns short	:1	00008000	When set to 1, there may be RTL Tables in this document
508	1FC	fNoTabForInd	uns long	:1	00000001	Compatibility option: when set to 1, don’t add automatic tab stop for hanging indent
		fNoSpaceRaiseLower	uns long	:1	00000002	Compatibility option: when set to 1, don't add extra space for raised/lowered characters
		fSuppressSpBfAfterPgBrk	uns long	:1	00000004	Compatibility option: when set to 1, suppress Space Before after a hard page or column break
		fWrapTrailSpaces	uns long	:1	00000008	Compatibility option: when set to 1, wrap trailing spaces to the next line
		fMapPrintTextColor	uns long	:1	00000010	Compatibility option: when set to 1, print colors as black on non-color printer
		fNoColumnBalance	uns long	:1	00000020	Compatibility option: when set to 1, don't balance columns for continuous section starts
		fConvMailMergeEsc	uns long	:1	00000040	Compatibility option: when set to 1, treat \" as "" in mail merge data sources

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
		fSuppressTopSpacing	uns long	:1	00000080	Compatibility option: when set to 1, suppress extra line spacing at top of page.
		fOrigWordTableRules	uns long	:1	00000100	Compatibility option: compatibility option: when 1, combine table borders like Word 5.x for the Macintosh
		fTransparentMetafiles	uns long	:1	00000200	Compatibility option: when set to 1, don't blank the area behind metafile pictures
		fShowBreaksInFrames	uns long	:1	00000400	Compatibility option: when set to 1, show hard page or column breaks in frames
		fSwapBordersFacingPgs	uns long	:1	00000800	Compatibility option: when set to 1, swap left and right borders n odd facing pages
		fLeaveBackslashAlone	uns long	:1	00001000	Compatibility option: when set to 1, do not convert backslash characters into yen signs
		fExpShRtn	uns long	:1	00002000	Compatibility option: when set to 1, expand character spaces on the line ending SHIFT+RETURN
		fDntULTrlSpc	uns long	:1	00004000	Compatibility option: when set to 1, don’t underline trailing spaces
		fDntBlnSbDbWid	uns long	:1	00008000	Compatibility option: when set to 1, don't balance SBCS and DBCS characters
		fSuppressTopSpacingMac5	uns long	:1	00010000	Compatibility option: when set to 1, suppress extra line spacing at the top of the page like Word 5.x for the Macintosh
		fTruncDxaExpand	uns long	:1	00020000	Compatibility option: when set to 1, expand/condense by whole number of points
		fPrintBodyBeforeHdr	uns long	:1	00040000	Compatibility option: when set to 1, print body text before header/footer
		fNoExtLeading	uns long	:1	00080000	Compatibility option: when set to 1, don't add leading space between rows of text
		fMakeSpaceForUL	uns long	:1	00100000	Compatibility option: when set to 1, add space for underlines.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
		fMWSmallCaps	uns long	:1	00200000	Compatibility option: when set to 1, use larger small caps like Word 5.x for the Macintosh
		f2ptExtLeadingOnly	uns long	:1	00400000	Compatibility option: suppress extra line spacing like WordPerfect
		fTruncFontHeight	uns long	:1	00800000	Compatibility option: when set to 1, truncate font height
		fSubOnSize	uns long	:1	01000000	Compatibility option: when set to 1, substitute fonts based on size.
		fLineWrapLikeWord6	uns long	:1	02000000	Compatibility option: when set to 1, lines wrap like Word 6.0
		fWW6BorderRules	uns long	:1	04000000	Compatibility option: when set to 1, use Word 6.0/95/97 border rules.
		fExactOnTop	uns long	:1	08000000	Compatibility option: when set to 1, don't center "exact line height" lines
		fExtraAfter	uns long	:1	10000000	Compatibility option: when set to 1, suppress extra line spacing at bottom of page
		fWPSpace	uns long	:1	20000000	Compatibility option: when set to 1, set the width of a space like WordPerfect 5
		fWPJust	uns long	:1	40000000	Compatibility option: when set to 1, do full justification like WordPerfect 6.x
		fPrintMet	uns long	:1	80000000	Compatibility option: when set to 1, use printer metrics to lay out the document
512	200	fSpLayoutLikeWW8	uns long	:1	00000001	Compatibility option: when set to 1, lay AutoShapes like Word 97
		fFtnLayoutLikeWW8	uns long	:1	00000002	Compatibility option: when set to 1, lay footnotes like Word 6.x/95/97.
		fDontUseHTMLParagraphAutoSpacing	uns long	:1	00000004	Compatibility option: when set to 1, don't use HTML paragraph auto spacing
		fDontAdjustLineHeightInTable	uns long	:1	00000008	Compatibility option: when set to 1, don't adjust line height in tables

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
		fForgetLastTabAlign	uns long	:1	00000010	Compatibility option: when set to 1, forget last tab alignment
		fUseAutospaceForFullWidthAlpha	uns long	:1	00000020	Compatibility option: when set to 1, use auto space like Word 95
		fAlignTablesRowByRow	uns long	:1	00000040	Compatibility option: when set to 1, align table rows independently
		fLayoutRawTableWidth	uns long	:1	00000080	Compatibility option: when set to 1, lay out tables with raw width
		fLayoutTableRowsApart	uns long	:1	00000100	Compatibility option: when set to 1, allow table rows to lay out apart
		fUseWord97LineBreakingRules	uns long	:1	00000200	Compatibility option: when set to 1, use Word 97 line breaking rules for East Asian text
		fDontBreakWrappedTables	uns long	:1	00000400	Compatibility option: Do not break wrapped tables across pages.
		fDontSnapToGridInCell	uns long	:1	00000800	Compatibility option: Do not snap text to grid while in a table with inline objects.
		fDontAllowFieldEndSelect	uns long	:1	00001000	Compatibility option: Select the entire field with the first or last character
		fApplyBreakingRules	uns long	:1	00002000	Compatibility option: Apply breaking rules
		fDontWrapTextWithPunct	uns long	:1	00004000	Compatibility option: Do not allow hanging punctuation with character grid
		fDontUseAsianBreakRules	uns long	:1	00008000	Compatibility option: Do not use Asian break rules for line breaks with character grid.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
		fUseWord2002TableStyleRules	uns long	:1	00010000	Compatibility option: Use the Word 2002 table style rules. Word 2002 places the top border of a column under the heading row, rather than above it as Word 2003 does. Word 2003 applies the top border of a column in a more intuitive place when there is a header row in the table. This new behavior also fixes an issue with shading not displaying correctly for cells using conditional formatting.
		fGrowAutofit	uns long	:1	00020000	Compatibility option: Allow tables set to "autofit to contents" to extend into the margins when in Print Layout. Word 2003 does not allow this by default.
		empty	uns long	:14	FFFC0000	Not used
516	204	empty	uns long	:32		Not used
520	208	empty	uns long	:32		Not used
524	20C	empty	uns long	:32		Not used
528	210	empty	uns long	:32		Not used
532	214	empty	uns long	:32		Not used
536	218	empty	uns long	:31		Not used
		private	uns long	:1		Not used
540	21C	verCompatPreW10	uns long	:16	0000FFFF	HTML I/O compatibility level
		fNoMargPgvwSaved	uns long	:1	00010000	Page view option
		fNoMargPgvwPag	uns long	:1	00020000	Page view option
		fWebViewPag	uns long	:1	00040000	Web View option

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
		fSeeDrawingsPag	uns long	:1	00080000	
		fBulletProofed	uns long	:1	00100000	this doc was produced by the document BulletProofer
		fCorrupted	uns long	:1	00200000	this doc was doctored by the Document Corrupter
		fSaveUim	uns long	:1	00400000	Save option: Embed linguistic in the doc
		fFilterPrivacy	uns long	:1	00800000	Save option: Remove personal information on save
		fInFReplaceNoRM	uns long	:1	01000000	we are under FReplace (and not just FReplaceRM)
		fSeenRepairs	uns long	:1	02000000	The user has seen the repairs made to the document
		fHasXML	uns long	:1	04000000	XML: The document has XML
		fSeeScriptAnchorsPag	uns long	:1	08000000	
		fValidateXML	uns long	:1	10000000	XML option: Validate XML on save
		fSaveIfInvalidXML	uns long	:1	20000000	XML option: Save the document even if the XML is invalid
		fShowXMLErrors	uns long	:1	40000000	XML option: Show any errors in the XML
		fAlwaysMergeEmptyNamespace	uns long	:1	80000000	we imported an XML file that had no namespace, so we have elements with no namespace and no schema
544	220	cpMaxListCacheMainDoc	CP (long)			
548	224	fDoNotEmbedSystemFont	uns short	:1	0001	Do not embed system fonts in this document
		fWordCompat	uns short	:1	0002	see fWord97Compat
		fLiveRecover	uns short	:1	0004	
		fEmbedFactoids	uns short	:1	0008	Embed smart tags in the document
		fFactoidXML	uns short	:1	0010	Save smart tags as XML properties
		fFactoidAllDone	uns short	:1	0020	Done processing smart tags

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
		fFolioPrint	uns short	:1	0040	Print option: Book fold
		fReverseFolio	uns short	:1	0080	Print option: Reverse book fold
		iTextLineEnding	uns short	:3	0700	
		fHideFcc	uns short	:1	0800	Do not keep track of formatting
		fAcetateShowMarkup	uns short	:1	1000	Track changes: show markup
		fAcetateShowAtn	uns short	:1	2000	Track changes: show annotations
		fAcetateShowInsDel	uns short	:1	4000	Track changes: show insertions and deletions
		fAcetateShowProps	uns short	:1	8000	Track changes: show formatting
550	226	istdTableDflt	uns long	:16		Default table style for the document
		verCompat	Uns long	:16		Internal: Version compatibility for save
554	22A	grfFmtFilter	Uns short			Internal: filter state for the Styles and Formatting Pane.
556	22C	iFolioPages	short			Book fold printing: sheets per booklet
558	22E	cpgText	CPG (Uns short)			
560	230	cpMinRMText	CP (long)			Revision mark CP info
564	234	cpMinRMFtn	CP (long)			Revision mark CP info
568	238	cpMinRMHdd	CP (long)			Revision mark CP info
572	23C	cpMinRMAtn	CP (long)			Revision mark CP info
576	240	cpMinRMEdn	CP (long)			Revision mark CP info
580	244	cpMinRMTxbx	CP (long)			Revision mark CP info
584	248	cpMinRMHdrTxbx	CP (long)			Revision mark CP info

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
588	24C	rsidRoot	RSID			
592	250	fTreatLockAtnAsReadOnly	uns long	:1	00000001	Document Protection: Treat lock for annotations as Read Only
		fStyleLock	uns long	:1	00000002	Document Protection: Style lockdown is turned on
		fAutoFmtOverride	uns long	:1	00000004	Document Protection: Allow AutoFormat to override style lockdown
		fRemoveWordML	uns long	:1	00000008	XML Option: Remove Word XML when saving; save only non-Word XML data.
		fApplyCustomXForm	uns long	:1	00000010	XML Option: Apply custom transform on Save
		fStyleLockEnforced	uns long	:1	00000020	Document Protection: Style lockdown is enforced
		fFakeLockAtn	uns long	:1	00000040	Document Protection: Simulate locked for annotations in older version when a document has style protection
		fIgnoreMixedContent	uns long	:1	00000080	XML Option: Ignore mixed content
		fShowPlaceholderText	uns long	:1	00000100	XML Option: Show placeholder text for all empty XML elements
		grf	uns long	:23	FFFFFE00	
596	254	fReadingModeInkLockDown	uns short	:1	0001	Reading mode: ink lock down
		fAcetateShowInkAtn	uns short	:1	0002	Track changes: Show ink annotations
		fFilterDttm	uns short	:1	0004	Filter date and time
		fEnforceDocProt	uns short	:1	0008	Enforce document protection
		iDocProtCur	uns short	:3	0070	Doc protection level: 0 Protect for track changes 1 Comment protection 2 Form protection 3 Read Only
		fDispBkSpSaved	uns short	:1	0080	

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
		fSpare	uns short	:8	FF00	Not used
598	256	dxaPageLock	XA			Reading Layout page size lockdown
600	258	dyaPageLock	YA			Reading Layout page size lockdown
602	25A	pctFontLock	int			Reading Layout font lockdown
606	25E	grfitbid	uchar			
607	25F		uchar	:8		Not used
608	260	ilfoMacAtCleanup	ushort			Number of LFOs when CleanupLists last attempted cleaning

For this expanded DOP, the size is 610 bytes.

Summary of nFib values:

WinWord 1.0 = 33

WinWord 2.0 = 45

WinWord 6.0c for 16bit = 101

Word 6/32 bit = 104

Word 95 = 104

Word 97 = 193

Word 2000 = 217

Word 2002 = 257

Word 2003 = 268

Word 2007 = 274

Number Format Table

nfc value	Numbering scheme
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)
10	Kanji numbering without the digit character (dbnum1).
11	Kanji numbering with the digit character (dbnum2).
12	46 phonetic Katakana characters in "aiueo" order (aiueo).
13	46 phonetic katakana characters in "iroha" order (iroha).
14	Double Byte character

nfc value	Numbering scheme
15	Single Byte character
16	Kanji numbering 3 (dbnum3).
17	Kanji numbering 4 (dbnum4).
18	Circle numbering (circlenum).
19	Double-byte Arabic numbering
20	46 phonetic double-byte Katakana characters (*aiueo*dbchar).
21	46 phonetic double-byte katakana characters (*iroha*dbchar).
22	Arabic with leading zero (01, 02, 03, ..., 10, 11)
23	Bullet (no number at all)
24	Korean numbering 2 (ganada).
25	Korean numbering 1 (chosung).
26	Chinese numbering 1 (gb1).
27	Chinese numbering 2 (gb2).
28	Chinese numbering 3 (gb3).
29	Chinese numbering 4 (gb4).
30	Chinese Zodiac numbering 1
31	Chinese Zodiac numbering 2
32	Chinese Zodiac numbering 3
33	Taiwanese double-byte numbering 1
34	Taiwanese double-byte numbering 2
35	Taiwanese double-byte numbering 3
36	Taiwanese double-byte numbering 4
37	Chinese double-byte numbering 1
38	Chinese double-byte numbering 2
39	Chinese double-byte numbering 3
40	Chinese double-byte numbering 4
41	Korean double-byte numbering 1
42	Korean double-byte numbering 2
43	Korean double-byte numbering 3
44	Korean double-byte numbering 4
45	Hebrew non-standard decimal
46	Arabic Alif Ba Tah
47	Hebrew Biblical standard
48	Arabic Abjad style
49	Hindi vowels
50	Hindi consonants
51	Hindi numbers
52	Hindi descriptive (cardinals)
53	Thai letters
54	Thai numbers
55	Thai descriptive (cardinals
56	Vietnamese descriptive (cardinals)
57	Page Number format - # -
58	Lower case Russian alphabet

nfc value	Numbering scheme
59	Upper case Russian alphabet

Drawing Object Grid (DOGRID)

The drawing object grid is East Asian only, and it sets up a grid in which East Asian characters are displayed (one character per grid square).

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
0	0	xaGrid	short			x-coordinate of the upper left-hand corner of the grid
2	2	yaGrid	short			y-coordinate of the upper left-hand corner of the grid
4	4	dxaGrid	short			Width of each grid square
6	6	dyaGrid	short			Height of each grid square
8	8	dyGridDisplay	short	:7	007F	The number of grid squares (in the y direction) between each gridline drawn on the screen. 0 means don't display any gridlines in the y direction.
		fTurnItOff	short	:1	0080	Suppress display of gridlines
		dxGridDisplay	short	:7	7F00	The number of grid squares (in the x direction) between each gridline drawn on the screen. 0 means don't display any gridlines in the y direction.
		fFollowMargins	short	:1	8000	If true, the grid will start at the left and top margins and ignore xaGrid and yaGrid

cbDOGRID (count of bytes of DOGRID) is 10 bytes (decimal), A bytes (hex).

Document Typography Info (DOPTYPOGRAPHY)

These options are East Asian only, and are accessible through the Typography tab of the Tools/Options dialog.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
0	0	fKerningPunct	short	:1	00000001	True if we're kerning punctuation
		iJustification	short	:2	00000006	Kinsoku method of justification: 0 = always expand 1 = compress punctuation 2 = compress punctuation and kana
		iLevelOfKinsoku	short	:2	00000018	Level of Kinsoku: 0 = Level 1 1 = Level 2 2 = Custom
		f2on1	short	:1	00000020	2-page-on-1 feature is turned on
		fOldDefineLineBasedOnGrid	short	:1	5F	Old East Asian feature
		iCustomKsu	short	:3	380	Custom Kinsoku
		fJapaneseUseLevel2	short	:1	400	When set to 1, use strict (level 2) Kinsoku rules
		reserved	short	:5	F800	Reserved

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
2	2	cchFollowingPunct	short			Length of rgxchFPunct
4	4	cchLeadingPunct	short			Length of rgxchLPunct
6	6	rgxchFPunct	XCHAR [101]			Array of characters that should never appear at the start of a line
208	D0	rgxchLPunct	XCHAR [51]			Array of characters that should never appear at the end of a line

cbDOPTYPOGRAPHY (count of bytes of DOPTYPOGRAPHY) is 310 bytes (decimal), 136 (hex).

Field Descriptor (FLD)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
0	0	ch	char	:5	1F	Type of field boundary the FLD describes: 19 field begin mark 20 field separator mark 21 field end mark
			char	:3	E0	Reserved
						Variant used when fld.ch == 19 (field begin mark)
1	1	flt	char			Field type (see flt table below)
						Variant used when fld.ch == 21 (field end mark)
1	1	fDiffer	char	:1	01	Ignored for saved file
		fZombieEmbed	char	:1	02	==1 when result still believes this field is an EMBED or LINK field
		fResultDirty	char	:1	04	==1 when user has edited or formatted the result. == 0 otherwise.
		fResultEdited	char	:1	08	==1 when user has inserted text into or deleted text from the result
		fLocked	char	:1	10	==1 when field is locked from recalculation
		fPrivateResult	char	:1	20	==1 whenever the result of the field is never to be shown
		fNested	char	:1	40	==1 when field is nested within another field
		fHasSep	char	:1	80	==1 when field has a field separator

flt value	live/dead	Field type
1		Unknown keyword
2	live	Possible bookmark (syntax matches bookmark name)
3	live	Bookmark reference
4	dead	Index entry
5	live	Footnote reference
6	live	Set command (for Print Merge)
7	live	If command (for Print Merge)
8	live	Create index
9	dead	Table of contents entry

flt value	live/dead	Field type
10	live	Style reference
11	dead	Document reference
12	live	Sequence mark
13	live	Create table-of-contents
14	live	Quote Info variable
15	live	Quote Title variable
16	live	Quote Subject variable
17	live	Quote Author variable
18	live	Quote Keywords variable
19	live	Quote Comments variable
20	live	Quote Last Revised By variable
21	live	Quote Creation Date variable
22	live	Quote Revision Date variable
23	live	Quote Print Date variable
24	live	Quote Revision Number variable
25	live	Quote Edit Time variable
26	live	Quote Number of Pages variable
27	live	Quote Number of Words variable
28	live	Quote Number of Characters variable
29	live	Quote File Name variable
30	live	Quote Document Template Name variable
31	live	Quote Current Date variable
32	live	Quote Current Time variable
33	live	Quote Current Page variable
34	live	Evaluate expression
35	live	Insert literal text
36	live	Include command (Print Merge)
37	live	Page reference
38	live	Ask command (Print Merge)
39	live	Fill-in command to display prompt (Print Merge)
40	live	Data command (Print Merge)
41	live	Next command (Print Merge)
42	live	NextIf command (Print Merge)
43	live	SkipIf (Print Merge)
44	live	Inserts number of current Print Merge record
45	live	DDE reference
46	live	DDE automatic reference
47	live	Inserts Glossary Entry
48	live	Sends characters to printer without translation
49	live	Formula definition
50	live	Goto Button
51	live	Macro Button
52	live	Insert auto numbering field in outline format

flt value	live/dead	Field type
53	live	Insert auto numbering field in legal format
54	live	Insert auto numbering field in Arabic number format
55	live	Reads a TIFF file
56	live	Link
57	live	Symbol
58	live	Embedded Object
59	live	Merge fields
60	live	User Name
61	live	User Initial
62	live	User Address
63	live	Bar code
64	live	Document variable
65	live	Section
66	live	Section pages
67	live	Include Picture
68	live	Include Text
69	live	File Size
70	live	Form Text Box
71	live	Form Check Box
72	live	Note Reference
73	live	Create Table of Authorities
74	dead	Mark Table of Authorities Entry
75	live	Merge record sequence number
76	either	Macro
77	dead	Private
78	live	Insert Database
79	live	Autotext
80	live	Compare two values
81	live	Plug-in module private
82	live	Subscriber
83	live	Form List Box
84	live	Advance
85	live	Document property
86	live	
87	live	OCX
88	live	Hyperlink
89	live	AutoTextList
90	live	List element
91	live	HTML control
92	live	Bidi Outline
93	live	Address Block
94	live	Greeting Line
95	live	Pseudo-inline shape

Since dead fields have no entry in the plcffld, the string in the field code must be used to determine the field type. All versions of Word '97 use English field code strings, except French, German, and Spanish versions of Word. The strings for all languages for all possible dead fields are listed below.

flt value	English string	French string	German string	Spanish string	Field type
4	**XE**	**EX**	**XE**	**E**	Index entry
9	**TC**	**TE**	**INHALT**	**TC**	Table of contents entry
11	**RD**	**RD**	**RD**	**RD**	Document reference
74	**TA**	**TA**	**TA**	**TA**	Table of authorities entry
76					Macro
77	**PRIVATE**	**PRIVE**	**PRIVATE**	**PRIVATESPA**	Private

File Shape Address (FSPA)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
0	0	spid	long			Shape Identifier. Used in conjunction with the office art data (found via fcDggInfo in the FIB) to find the actual data for this shape.
4	4	xaLeft	xa			Left of rectangle enclosing shape relative to the origin of the shape
8	8	yaTop	ya			Top of rectangle enclosing shape relative to the origin of the shape
12	C	xaRight	xa			Right of rectangle enclosing shape relative to the origin of the shape
16	10	yaBottom	ya			Bottom of the rectangle enclosing shape relative to the origin of the shape
20	14	fHdr	uns short	:1	0001	1 in the undo doc when shape is from the header doc, 0 otherwise (undefined when not in the undo doc)
		bx	uns short	:2	0006	X position of shape relative to anchor CP 0 relative to page margin 1 relative to top of page 2 relative to text (column for horizontal text; paragraph for vertical text) 3 reserved for future use
		by	uns short	:2	0018	Y position of shape relative to anchor CP 0 relative to page margin 1 relative to top of page 2 relative to text (paragraph for horizontal text; column for vertical text)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
		wr	uns short	:4	01E0	Text wrapping mode 0 like 2, but doesn't require absolute object 1 no text next to shape 2 wrap around absolute object 3 wrap as if no object present 4 wrap tightly around object 5 wrap tightly, but allow holes 6-15 reserved for future use
		wrk	uns short	:4	1E00	Text wrapping mode type (valid only for wrapping modes 2 and 4 0 wrap both sides 1 wrap only on left 2 wrap only on right 3 wrap only on largest side
		fRcaSimple	uns short	:1	2000	When set, temporarily overrides bx, by, forcing the xaLeft, xaRight, yaTop, and yaBottom fields to all be page relative.
		fBelowText	uns short	:1	4000	1 shape is below text 0 shape is above text
		fAnchorLock	uns short	:1	8000	1 anchor is locked 0 anchor is not locked
22	16	cTxbx	long			Count of textboxes in shape (undo doc only)

cbFSPA (count of bytes of FSPA) is 26 (decimal), 1A (hex).

Font Family Name (FFN)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
0	0	cbFfnM1	uns char			Total length of FFN - 1.
1	1	prq	uns char	:2	03	Pitch request
		fTrueType	uns char	:1	04	When 1, font is a TrueType font
			uns char	:1	08	Reserved
		ff	uns char	:3	70	Font family id
			uns char	:1	80	Reserved
2	2	wWeight	short			Base weight of font
4	4	chs	uns char			Character set identifier
5	5	ixchSzAlt	uns char			Index into ffn.szFfn to the name of the alternate font
6	6	panose	PANOSE			
16	10	fs	FONTSIGNATURE			

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
40	28	xszFfn	XCHAR[]			Zero terminated string that records name of font. Possibly followed by a second xsz which records the name of an alternate font to use if the first named font does not exist on this system. Maximal size of xszFfn is 65 characters.

File Information Block (FIB)

In Word version 8, the FIB is reorganized to make future extension easier, and to make it easier to make backward compatible file format changes. The FIB now consists of four substructures: the header and three arrays. The FIB header, is unchanged from past versions. The second part is an array of 16-bit "shorts", most of which were present in earlier versions in different locations. The third part is an array of 32-bit longs, many of which were scattered through the previous version FIB. Finally, there is an array of FC/LCB pairs, which were divided into several disjoint arrays in the previous FIB. Future versions of Word will add entries to the three arrays, so readers of the FIB must be careful to skip over any entries in each array that were not present in the version for which the reader was designed. Writers of the FIB must write exactly as many entries as was defined for the nFib value they put in the FIB.

The FIBFCLCB structure, used in an array in the FIB:

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
0	0x0000	Fc	long			File position where data begins.	
4	0x0004	Lcb	ulong			Size of data. Ignore fc if lcb is zero.	

The FCPGDOLD structure, referenced in the FIB, used internally by Word:

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
0	0x0000	fcPgd	long			File position where data begins.	
4	0x0004	lcbPgd	ulong			Size of data. Ignore fc if lcb is zero.	
8	0x0008	fcBkd	long			File position where data begins.	
12	0x000C	lcbBkd	ulong			Size of data. Ignore fc if lcb is zero.	

The FCPGD structure, referenced in the FIB, used internally by Word. This modified version of the above structure was introduced in Word 2003:

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
0	0x0000	fcPgd	long			File position where data begins.	Word 2003
4	0x0004	lcbPgd	ulong			Size of data. Ignore fc if lcb is zero.	Word 2003
8	0x0008	fcBkd	long			File position where data begins.	Word 2003
12	0x000C	lcbBkd	ulong			Size of data. Ignore fc if lcb is zero.	Word 2003
16	0x0010	fcAfd	FC			File position where data begins.	Word 2003

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
20	0x0014	lcbAfd	ulong			Size of data. Ignore fc if lcb is zero.	Word 2003

The FIB structure itself:

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
0	0x0000	Fibh	FIBH			Beginning of the FIB header	Word 97
0	0x0000	wIdent	ushort			Magic number	Word 97
2	0x0002	nFib	ushort			FIB version written. This will be >= 101 for all Word 6.0 for Windows and after documents.	Word 97
4	0x0004	nProduct	ushort			Product version written by	Word 97
6	0x0006	Lid	ushort			Language stamp --localized version In pre-WinWord 2.0 files this value was the nLocale. If value is < 999, then it is the nLocale, otherwise it is the lid.	Word 97
8	0x0008	pnNext	short				Word 97
10	0x000A	fDot	ushort	:1	0x0001	Set if this document is a template	Word 97
		fGlsy	ushort	:1	0x0002	Set if this document is a glossary	Word 97
		fComplex	ushort	:1	0x0004	When 1, file is in complex, fast-saved format.	Word 97
		fHasPic	ushort	:1	0x0008	Set if file contains 1 or more pictures	Word 97
		cQuickSaves	ushort	:4	0x00F0	Count of times file was quick saved	Word 97
		fEncrypted	ushort	:1	0x0100	Set if file is encrypted	Word 97
		fWhichTblStm	ushort	:1	0x0200	When 0, this fib refers to the table stream named "0Table", when 1, this fib refers to the table stream named "1Table". Normally, a file will have only one table stream, but under unusual circumstances a file may have table streams with both names. In that case, this flag must be used to decide which table stream is valid.	Word 97
		fReadOnlyRecommended	ushort	:1	0x0400	Set when user has recommended that file be read read-only	Word 97
		fWriteReservation	ushort	:1	0x0800	Set when file owner has made the file write reserved	Word 97
		fExtChar	ushort	:1	0x1000	Set when using extended character set in file	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
		fLoadOverride	ushort	:1	0x2000	REVIEW	Word 97
		fFarEast	ushort	:1	0x4000	REVIEW	Word 97
		fCrypto	ushort	:1	0x8000	REVIEW	Word 97
12	0x000C	nFibBack	Ushort			This file format is compatible with readers that understand nFib at or above this value.	Word 97
14	0x000E	lKey				File encrypted key, only valid if fEncrypted	Word 97
18	0x0012	Envr	Uchar			Environment in which file was created 0 created by Word for Windows 1 created by Word for the Macintosh	Word 97
19	0x0013	fMac	Uchar	:1	0x01	When 1, this file was last saved in the Macintosh environment	Word 97
		fEmptySpecial	Uchar	:1	0x02		Word 97
		fLoadOverridePage	Uchar	:1	0x04		Word 97
		fFutureSavedUndo	Uchar	:1	0x08		Word 97
		fWord97Saved	Uchar	:1	0x10		Word 97
		fSpare0	Uchar	:3	0xFE		Word 97
20	0x0014	Chs	Ushort			Default extended character set id for text in document stream. (overridden by chp.chse) 0 by default characters in doc stream should be interpreted using the ANSI character set used by Windows 256 characters in doc stream should be interpreted using the Macintosh character set.	Word 97
22	0x0016	chsTables	Ushort			Default extended character set id for text in internal data structures 0 by default characters stored in internal data structures should be interpreted using the ANSI character set used by Windows 256 characters stored in internal data structures should be interpreted using the Macintosh character set.	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
24	0x0018	fcMin	Long			File offset of first character of text. In non-complex files a CP can be transformed into an FC by the following transformation: fc = cp + fib.fcMin.	Word 97
28	0x001C	fcMac	Long			File offset of last character of text in document text stream + 1	Word 97
32	0x0020	Csw	Ushort			Count of fields in the array of "shorts"	Word 97
34	0x0022	Rgsw				Beginning of the array of shorts	Word 97
34	0x0022	wMagicCreated				Unique number identifying the file's creator. 0x6A62 is the creator ID for Word and is reserved. Other creators should choose a different value.	Word 97
36	0x0024	wMagicRevised				Identifies the file's last modifier	Word 97
38	0x0026	wMagicCreatedPrivate				Private data	Word 97
40	0x0028	wMagicRevisedPrivate				Private data	Word 97
42	0x002A	pnFbpChpFirst_W6	Short			Not used	Word 97
44	0x002C	pnChpFirst_W6	Short			Not used	Word 97
46	0x002E	cpnBteChp_W6	Short			Not used	Word 97
48	0x0030	pnFbpPapFirst_W6	Short			Not used	Word 97
50	0x0032	pnPapFirst_W6	Short			Not used	Word 97
52	0x0034	cpnBtePap_W6	Short			Not used	Word 97
54	0x0036	pnFbpLvcFirst_W6	Short			Not used	Word 97
56	0x0038	pnLvcFirst_W6	Short			Not used	Word 97
58	0x003A	cpnBteLvc_W6	Short			Not used	Word 97
60	0x003C	lidFE	Short			Language id if document was written by East Asian version of Word (i.e. FIB.fFarEast is on)	Word 97
62	0x003E	Clw	Ushort			Number of fields in the array of longs	Word 97
64	0x0040	Rglw				Beginning of the array of longs	Word 97
64	0x0040	cbMac	Long			File offset of last byte written to file + 1	Word 97
68	0x0044	lProductCreated				Contains the build date of the creator. 10695 indicates the creator program was compiled on Jan 6, 1995.	Word 97
72	0x0048	lProductRevised				Contains the build date of the file's last modifier	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
76	0x004C	ccpText	Long			Length of main document text stream 1	Word 97
80	0x0050	ccpFtn	Long			Length of footnote subdocument text stream	Word 97
84	0x0054	ccpHdd	Long			Length of header subdocument text stream	Word 97
88	0x0058	ccpMcr	Long			Length of macro subdocument text stream, which should now always be 0	Word 97
92	0x005C	ccpAtn	Long			Length of annotation subdocument text stream	Word 97
96	0x0060	ccpEdn	Long			Length of endnote subdocument text stream	Word 97
100	0x0064	ccpTxbx	Long			Length of textbox subdocument text stream	Word 97
104	0x0068	ccpHdrTxbx	Long			Length of header textbox subdocument text stream	Word 97
108	0x006C	pnFbpChpFirst	Long			When there was insufficient memory for Word to expand the plcfbte at save time, the plcfbte is written to the file in a linked list of 512-byte pieces starting with this pn	Word 97
112	0x0070	pnChpFirst	Long			The page number of the lowest numbered page in the document that records CHPX FKP information	Word 97
116	0x0074	cpnBteChp	Long			Count of CHPX FKPs recorded in file. In non-complex files if the number of entries in the plcfbteChpx is less than this, the plcfbteChpx is incomplete	Word 97
120	0x0078	pnFbpPapFirst	Long			When there was insufficient memory for Word to expand the plcfbte at save time, the plcfbte is written to the file in a linked list of 512-byte pieces starting with this pn	Word 97
124	0x007C	pnPapFirst	Long			The page number of the lowest numbered page in the document that records PAPX FKP information	Word 97
128	0x0080	cpnBtePap	Long			Count of PAPX FKPs recorded in file. In non-complex files if the number of entries in the plcfbtePapx is less than this, the plcfbtePapx is incomplete.	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
132	0x0084	pnFbpLvcFirst	Long			When there was insufficient memory for Word to expand the plcfbte at save time, the plcfbte is written to the file in a linked list of 512-byte pieces starting with this pn	Word 97
136	0x0088	pnLvcFirst	Long			The page number of the lowest numbered page in the document that records LVC FKP information	Word 97
140	0x008C	cpnBteLvc	Long			Count of LVC FKPs recorded in file. In non-complex files if the number of entries in the plcfbtePapx is less than this, the plcfbtePapx is incomplete.	Word 97
144	0x0090	fcIslandFirst	Long				Word 97
148	0x0094	fcIslandLim	Long				Word 97
152	0x0098	Cfclcb	Ushort			Number of fields in the array of FC/LCB pairs	Word 97
154	0x009A	Rgfclcb				Beginning of array of FC/LCB pairs	Word 97
154	0x009A	fcStshfOrig	Long			File offset of original allocation for STSH in table stream. During fast save Word will attempt to reuse this allocation if STSH is small enough to fit.	Word 97
158	0x009E	lcbStshfOrig	Ulong			Count of bytes of original STSH allocation	Word 97
162	0x00A2	fcStshf	Long			Offset of STSH in table stream	Word 97
166	0x00A6	lcbStshf	Ulong			Count of bytes of current STSH allocation	Word 97
170	0x00AA	fcPlcffndRef	Long			Offset in table stream of footnote reference PLCF of FRD structures. CPs in PLC are relative to main document text stream and give location of footnote references.	Word 97
174	0x00AE	lcbPlcffndRef	Ulong			Count of bytes of footnote reference PLC== 0 if no footnotes defined in document	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
178	0x00B2	fcPlcffndTxt	Long			Offset in table stream of footnote text PLC. CPs in PLC are relative to footnote subdocument text stream and give location of beginnings of footnote text for corresponding references recorded in plcffndRef. No structure is stored in this plc. There will just be n+1 FC entries in this PLC when there are n footnotes.	Word 97
182	0x00B6	lcbPlcffndTxt	Ulong			Count of bytes of footnote text PLC. == 0 if no footnotes defined in document.	Word 97
186	0x00BA	fcPlcfandRef	Long			Offset in table stream of annotation reference ATRDPre10 PLC. The CPs recorded in this PLC give the offset of annotation references in the main document.	Word 97
190	0x00BE	lcbPlcfandRef	Ulong			Count of bytes of annotation reference PLC	Word 97
194	0x00C2	fcPlcfandTxt	Long			Offset in table stream of annotation text PLC. The CPs recorded in this PLC give the offset of the annotation text in the annotation sub document corresponding to the references stored in the plcfandRef. There is a 1-to-1 correspondence between entries recorded in the plcfandTxt and the plcfandRef. No structure is stored in this PLC.	Word 97
198	0x00C6	lcbPlcfandTxt	Ulong			Count of bytes of the annotation text PLC	Word 97
202	0x00CA	fcPlcfsed	Long			Offset in table stream of section descriptor SED PLC. CPs in PLC are relative to main document.	Word 97
206	0x00CE	lcbPlcfsed	Ulong			Count of bytes of section descriptor PLC	Word 97
210	0x00D2	fcPlcpad	Long			No longer used	Word 97
214	0x00D6	lcbPlcpad	Ulong			No longer used	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
218	0x00DA	fcPlcfphe	Long			Offset in table stream of PHE PLC of paragraph heights. CPs in PLC are relative to main document text stream. Only written for files in complex format. Should not be written by third party creators of Word files.	Word 97
222	0x00DE	lcbPlcfphe	Ulong			Count of bytes of paragraph height PLC. ==0 when file is non-complex.	Word 97
226	0x00E2	fcSttbfglsy	Long			Offset in table stream of glossary string table. This table consists of Pascal-style strings (strings stored prefixed with a length byte) concatenated one after another.	Word 97
230	0x00E6	lcbSttbfglsy	Ulong			Count of bytes of glossary string table. == 0 for non-glossary documents. !=0 for glossary documents.	Word 97
234	0x00EA	fcPlcfglsy	Long			Offset in table stream of glossary PLC. CPs in PLC are relative to main document and mark the beginnings of glossary entries and are in 1-1 correspondence with entries of sttbfglsy. No structure is stored in this PLC. There will be n+1 FC entries in this PLC when there are n glossary entries.	Word 97
238	0x00EE	lcbPlcfglsy	Ulong			Count of bytes of glossary PLC. == 0 for non-glossary documents. !=0 for glossary documents.	Word 97
242	0x00F2	fcPlcfhdd	Long			Byte offset in table stream of header HDD PLC. CPs are relative to header subdocument and mark the beginnings of individual headers in the header subdocument. No structure is stored in this PLC. There will be n+1 FC entries in this PLC when there are n headers stored for the document.	Word 97
246	0x00F6	lcbPlcfhdd	Ulong			Count of bytes of header PLC. == 0 if document contains no headers.	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
250	0x00FA	fcPlcfbteChpx	Long			Offset in table stream of character property bin table.PLC. FCs in PLC are file offsets in the main stream. Describes text of main document and all subdocuments.	Word 97
254	0x00FE	lcbPlcfbteChpx	Ulong			Count of bytes of character property bin table PLC	Word 97
258	0x0102	fcPlcfbtePapx	Long			Offset in table stream of paragraph property bin table.PLC. FCs in PLC are file offsets in the main stream. Describes text of main document and all subdocuments.	Word 97
262	0x0106	lcbPlcfbtePapx	Ulong			Count of bytes of paragraph property bin table PLC	Word 97
266	0x010A	fcPlcfsea	Long			Offset in table stream of PLC reserved for private use. The SEA is 6 bytes long.	Word 97
270	0x010E	lcbPlcfsea	Ulong			Count of bytes of private use PLC	Word 97
274	0x0112	fcSttbfffn	Long			Offset in table stream of font information STTBF. The sttbfffn is a STTBF where is string is actually an FFN structure. The nth entry in the STTBF describes the font that will be displayed when the chp.ftc for text is equal to n. See the FFN file structure definition.	Word 97
278	0x0116	lcbSttbfffn	Ulong			Count of bytes in sttbfffn	Word 97
282	0x011A	fcPlcffldMom	Long			Offset in table stream to the FLD PLC of field positions in the main document. The CPs point to the beginning CP of a field, the CP of field separator character inside a field and the ending CP of the field. A field may be nested within another field. 20 levels of field nesting are allowed.	Word 97
286	0x011E	lcbPlcffldMom	Ulong			Count of bytes in plcffldMom	Word 97
290	0x0122	fcPlcffldHdr	Long			Offset in table stream to the FLD PLC of field positions in the header subdocument	Word 97
294	0x0126	lcbPlcffldHdr	Ulong			Count of bytes in plcffldHdr	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
298	0x012A	fcPlcffldFtn	Long			Offset in table stream to the FLD PLC of field positions in the footnote subdocument	Word 97
302	0x012E	lcbPlcffldFtn	Ulong			Count of bytes in plcffldFtn	Word 97
306	0x0132	fcPlcffldAtn	Long			Offset in table stream to the FLD PLC of field positions in the annotation subdocument	Word 97
310	0x0136	lcbPlcffldAtn	Ulong			Count of bytes in plcffldAtn	Word 97
314	0x013A	fcPlcffldMcr	Long			No longer used	Word 97
318	0x013E	lcbPlcffldMcr	Ulong			No longer used	Word 97
322	0x0142	fcSttbfbkmk	Long			Offset in table stream of the STTBF that records bookmark names in the main document	Word 97
326	0x0146	lcbSttbfbkmk	Ulong			Count of bytes in Sttbfbkmk	Word 97
330	0x014A	fcPlcfbkf	Long			Offset in table stream of the PLCF that records the beginning CP offsets of bookmarks in the main document. See BKF structure definition.	Word 97
334	0x014E	lcbPlcfbkf	Ulong			Count of bytes in Plcfbkf	Word 97
338	0x0152	fcPlcfbkl	Long			Offset in table stream of the PLCF that records the ending CP offsets of bookmarks recorded in the main document. No structure is stored in this PLCF.	Word 97
342	0x0156	lcbPlcfbkl	Ulong			Count of bytes in Plcfbkl	Word 97
346	0x015A	fcCmds	Long			Offset in table stream of the macro commands. These commands are private and undocumented.	Word 97
350	0x015E	lcbCmds	Ulong			Count of bytes of the data above.	Word 97
354	0x0162	fcPlcmcr	Long			No longer used	Word 97
358	0x0166	lcbPlcmcr	Ulong			No longer used	Word 97
362	0x016A	fcSttbfmcr	Long			No longer used	Word 97
366	0x016E	lcbSttbfmcr	Ulong			No longer used	Word 97
370	0x0172	fcPrDrvr	long			Offset in table stream of the printer driver information (names of drivers, port, etc.)	Word 97
374	0x0176	lcbPrDrvr	ulong			Count of bytes of the printer driver information (names of drivers, port, etc.)	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
378	0x017A	fcPrEnvPort	long			Offset in table stream of the print environment in portrait mode	Word 97
382	0x017E	lcbPrEnvPort	ulong			Count of bytes of the print environment in portrait mode	Word 97
386	0x0182	fcPrEnvLand	long			Offset in table stream of the print environment in landscape mode	Word 97
390	0x0186	lcbPrEnvLand	ulong			Count of bytes of the print environment in landscape mode	Word 97
394	0x018A	fcWss	long			Offset in table stream of Window Save State data structure. WSS contains dimensions of document's main text window and the last selection made by Word user.	Word 97
398	0x018E	lcbWss	ulong			Count of bytes of WSS. ==0 if unable to store the window state. Should not be written by third party creators of Word files.	Word 97
402	0x0192	fcDop	long			Offset in table stream of document property data structure	Word 97
406	0x0196	lcbDop	ulong			Count of bytes of document properties	Word 97
410	0x019A	fcSttbfAssoc	long			Offset in table stream of STTBF of associated strings. The strings in this table specify document summary info and the paths to special documents related to this document. See documentation of the STTBFASSOC.	Word 97
414	0x019E	lcbSttbfAssoc	ulong			Count of bytes in SttbfAssoc	Word 97
418	0x01A2	fcClx	long			Offset in table stream of beginning of information for complex files. Consists of an encoding of all of the prms quoted by the document followed by the plcpcd (piece table) for the document.	Word 97
422	0x01A6	lcbClx	ulong			Count of bytes of complex file information == 0 if file is non-complex.	Word 97
426	0x01AA	fcPlcfpgdFtn	long			Not used	Word 97
430	0x01AE	lcbPlcfpgdFtn	ulong			Not used	Word 97
434	0x01B2	fcAutosaveSource	long			Offset in table stream of the name of the original file. fcAutosaveSource and cbAutosaveSource should both be 0 if auto save is off.	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
438	0x01B6	lcbAutosaveSource	ulong			Count of bytes of the name of the original file.	Word 97
442	0x01BA	fcGrpXstAtnOwners	long			Offset in table stream of group of strings recording the names of the owners of annotations stored in the document	Word 97
446	0x01BE	lcbGrpXstAtnOwners	ulong			Count of bytes of the group of strings	Word 97
450	0x01C2	fcSttbfAtnbkmk	long			Offset in table stream of the sttbf that records names of bookmarks for the annotation subdocument	Word 97
454	0x01C6	lcbSttbfAtnbkmk	ulong			Length in bytes of the sttbf that records names of bookmarks for the annotation subdocument	Word 97
458	0x01CA	fcPlcdoaMom	long			No longer used	Word 97
462	0x01CE	lcbPlcdoaMom	ulong			No longer used	Word 97
466	0x01D2	fcPlcdoaHdr	long			No longer used	Word 97
470	0x01D6	lcbPlcdoaHdr	ulong			No longer used	Word 97
474	0x01DA	fcPlcspaMom	long			Offset in table stream of the FSPA PLC for main document. == 0 if document has no Office Drawing objects.	Word 97
478	0x01DE	lcbPlcspaMom	ulong			Length in bytes of the FSPA PLC of the main document	Word 97
482	0x01E2	fcPlcspaHdr	long			Offset in table stream of the FSPA PLC for header document. == 0 if document has no Office Drawing objects.	Word 97
486	0x01E6	lcbPlcspaHdr	ulong			Length in bytes of the FSPA PLC of the header document.	Word 97
490	0x01EA	fcPlcfAtnbkf	long			Offset in table stream of BKF (bookmark first) PLC of the annotation subdocument	Word 97
494	0x01EE	lcbPlcfAtnbkf	ulong			Length in bytes of BKF (bookmark first) PLC of the annotation subdocument	Word 97
498	0x01F2	fcPlcfAtnbkl	long			Offset in table stream of BKL (bookmark last) PLC of the annotation subdocument	Word 97
502	0x01F6	lcbPlcfAtnbkl	ulong			Length in bytes of PLC marking the CP limits of the annotation bookmarks. No structure is stored in this PLC.	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
506	0x01FA	fcPms	long			Offset in table stream of PMS (Print Merge State) information block. This contains the current state of a print merge operation.	Word 97
510	0x01FE	lcbPms	ulong			Length in bytes of PMS. ==0 if no current print merge state. Should not be written by third party creators of Word files.	Word 97
514	0x0202	fcFormFldSttbs	long			Offset in table stream of form field sttbf which contains strings used in form field dropdown controls	Word 97
518	0x0206	lcbFormFldSttbs	ulong			Length in bytes of form field sttbf	Word 97
522	0x020A	fcPlcfendRef	long			Offset in table stream of endnote reference PLCF of FRD structures. CPs in PLCF are relative to main document text stream and give location of endnote references.	Word 97
526	0x020E	lcbPlcfendRef	ulong			Count of bytes of the plcfendRef	Word 97
530	0x0212	fcPlcfendTxt	long			Offset in table stream of plcfendRef which points to endnote text in the endnote document stream which corresponds with the plcfendRef. No structure is stored in this PLC.	Word 97
534	0x0216	lcbPlcfendTxt	ulong			Count of bytes for the above data	Word 97
538	0x021A	fcPlcffldEdn	long			Offset in table stream to FLD PLCF of field positions in the endnote subdocument	Word 97
542	0x021E	lcbPlcffldEdn	ulong			Count of bytes for the above data	Word 97
546	0x0222	fcPlcfpgdEdn	long			Not used	Word 97
550	0x0226	lcbPlcfpgdEdn	ulong			Not used	Word 97
554	0x022A	fcDggInfo	long			Offset in table stream of the Office Drawing object table data. The format of office Drawing object table data is found in a separate document.	Word 97
558	0x022E	lcbDggInfo	ulong			Length in bytes of the Office Drawing object table data	Word 97
562	0x0232	fcSttbfRMark	long			Offset in table stream to STTBF that records the author abbreviations for authors who have made revisions in the document	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
566	0x0236	lcbSttbfRMark	ulong			Count of bytes for the above data	Word 97
570	0x023A	fcSttbCaption	long			Offset in table stream to STTBF that records caption titles used in the document	Word 97
574	0x023E	lcbSttbCaption	ulong			Count of bytes for the above data	Word 97
578	0x0242	fcSttbAutoCaption	long			Offset in table stream to the STTBF that records the object names and indices into the caption STTBF for objects which get auto captions	Word 97
582	0x0246	lcbSttbAutoCaption	ulong			Count of bytes for the above data	Word 97
586	0x024A	fcPlcfwkb	long			Offset in table stream to WKB PLCF that describes the boundaries of contributing documents in a master document	Word 97
590	0x024E	lcbPlcfwkb	ulong			Count of bytes for the above data	Word 97
594	0x0252	fcPlcfspl	long			Offset in table stream of PLCF (of SPLS structures) that records spell check state	Word 97
598	0x0256	lcbPlcfspl	ulong			Count of bytes for the above data	Word 97
602	0x025A	fcPlcftxbxTxt	long			Offset in table stream of PLCF that records the beginning CP in the text box subdoc of the text of individual text box entries. No structure is stored in this PLCF	Word 97
606	0x025E	lcbPlcftxbxTxt	ulong			Count of bytes for the above data	Word 97
610	0x0262	fcPlcffldTxbx	long			Offset in table stream of the FLD PLCF that records field boundaries recorded in the textbox subdoc.	Word 97
614	0x0266	lcbPlcffldTxbx	ulong			Count of bytes for the above data	Word 97
618	0x026A	fcPlcfhdrtxbxTxt	long			Offset in table stream of PLCF that records the beginning CP in the header text box subdoc of the text of individual header text box entries. No structure is stored in this PLC.	Word 97
622	0x026E	lcbPlcfhdrtxbxTxt	ulong			Count of bytes for the above data	Word 97
626	0x0272	fcPlcffldHdrTxbx	long			Offset in table stream of the FLD PLCF that records field boundaries recorded in the header textbox subdoc.	Word 97
630	0x0276	lcbPlcffldHdrTxbx	ulong			Count of bytes for the above data	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
634	0x027A	fcStwUser	long			Macro user storage	Word 97
638	0x027E	lcbStwUser	ulong			Count of bytes for the above data	Word 97
642	0x0282	fcSttbttmbd	long			Offset in table stream of embedded true type font data	Word 97
646	0x0286	cbSttbttmbd	ulong			Count of bytes for the above data	Word 97
650	0x028A	fcCookieData	FC			NLCheck error handle will persist in file	Word 97
654	0x028E	lcbCookieData	ulong			Count of bytes for the above data	Word 97
658	0x0292	rgpgdbkdOldOld[3]	FCPGDOLD[3]			Index into the following three properties	Word 97
658	0x0292	fcpgdMotherOldOld	FCPGDOLD			Offsets in table stream of the PLF that records the page and break descriptors for the main text of the document	Word 97
674	0x02A2	fcpgdFtnOldOld	FCPGDOLD			Offsets in table stream of the PLF that records the page and break descriptors for the footnote text of the document	Word 97
690	0x02B2	fcpgdEdnOldOld	FCPGDOLD			Offsets in table stream of the PLF that records the page and break descriptors for the endnote text of the document	Word 97
706	0x02C2	fcSttbfIntlFld	long			Offset in table stream of the STTBF containing field keywords. This is only used in a small number of the international versions of Word. This field is no longer written to the file for nFib >= 167.	Word 97
710	0x02C6	lcbSttbfIntlFld	ulong			Always 0 for nFib>=167	Word 97
714	0x02CA	fcRouteSlip	long			Offset in table stream of a mailer routing slip	Word 97
718	0x02CE	lcbRouteSlip	ulong			Count of bytes for the above data	Word 97
722	0x02D2	fcSttbSavedBy	long			Offset in table stream of STTBF recording the names of the users who have saved this document alternating with the save locations	Word 97
726	0x02D6	lcbSttbSavedBy	ulong			Count of bytes for the above data	Word 97
730	0x02DA	fcSttbFnm	long			Offset in table stream of STTBF recording filenames of documents which are referenced by this document	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
734	0x02DE	lcbSttbFnm	ulong			Count of bytes for the above data	Word 97
738	0x02E2	fcPlcfLst	long			Offset in the table stream of list format information	Word 97
742	0x02E6	lcbPlcfLst	ulong			Count of bytes for the above data	Word 97
746	0x02EA	fcPlfLfo	long			Offset in the table stream of list format override information	Word 97
750	0x02EE	lcbPlfLfo	ulong			Count of bytes for the above data	Word 97
754	0x02F2	fcPlcftxbxBkd	long			Offset in the table stream of the textbox break table (a PLCF of BKDs) for the main document	Word 97
758	0x02F6	lcbPlcftxbxBkd	ulong			Count of bytes for the above data	Word 97
762	0x02FA	fcPlcftxbxHdrBkd	long			Offset in the table stream of the textbox break table (a PLCF of BKDs) for the header subdocument	Word 97
766	0x02FE	lcbPlcftxbxHdrBkd	ulong			Count of bytes for the above data	Word 97
770	0x0302	fcDocUndoWord9	long			Offset in main stream of undocumented undo / versioning data used pre Word10	Word 97
774	0x0306	lcbDocUndoWord9	ulong			Count of bytes for the above data	Word 97
778	0x030A	fcRgbuse	long			Offset in main stream of undocumented undo / versioning data	Word 97
782	0x030E	lcbRgbuse	ulong			Count of bytes for the above data	Word 97
786	0x0312	fcUsp	long			Offset in main stream of undocumented undo / versioning data	Word 97
790	0x0316	lcbUsp	ulong			Count of bytes for the above data	Word 97
794	0x031A	fcUskf	long			Offset in table stream of undocumented undo / versioning data	Word 97
798	0x031E	lcbUskf	ulong			Count of bytes for the above data	Word 97
802	0x0322	fcPlcupcRgbuse	long			Offset in table stream of undocumented undo / versioning data	Word 97
806	0x0326	lcbPlcupcRgbuse	ulong			Count of bytes for the above data	Word 97
810	0x032A	fcPlcupcUsp	long			Offset in table stream of undocumented undo / versioning data	Word 97
814	0x032E	lcbPlcupcUsp	ulong			Count of bytes for the above data	Word 97
818	0x0332	fcSttbGlsyStyle	long			Offset in table stream of string table of style names for glossary entries	Word 97
822	0x0336	lcbSttbGlsyStyle	ulong			Count of bytes for the above data	Word 97

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
826	0x033A	fcPlgosl	long			Offset in table stream of undocumented grammar options PL	Word 97
830	0x033E	lcbPlgosl	ulong			Count of bytes for the above data	Word 97
834	0x0342	fcPlcocx	long			Offset in table stream of undocumented ocx data	Word 97
838	0x0346	lcbPlcocx	ulong			Count of bytes for the above data	Word 97
842	0x034A	fcPlcfbteLvc	long			Offset in table stream of character property bin table.PLC. FCs in PLC are file offsets. Describes text of main document and all subdocuments.	Word 97
846	0x034E	lcbPlcfbteLvc	ulong			Count of bytes for the above data	Word 97
850	0x0352	ftModified	FILETIME				Word 97
850	0x0352	dwLowDateTime	ulong				Word 97
854	0x0356	dwHighDateTime	ulong				Word 97
858	0x035A	fcPlcflvcPre10	long			Offset in table stream of LVC PLCF used pre Word10	Word 97
862	0x035E	lcbPlcflvcPre10	ulong			Size of LVC PLCF, ==0 for non-complex files	Word 97
866	0x0362	fcPlcasumy	long			Offset in table stream of autosummary ASUMY PLCF.	Word 97
870	0x0366	lcbPlcasumy	ulong			Count of bytes for the above data	Word 97
874	0x036A	fcPlcfgram	long			Offset in table stream of PLCF (of SPLS structures) which records grammar check state	Word 97
878	0x036E	lcbPlcfgram	ulong			Count of bytes for the above data	Word 97
882	0x0372	fcSttbListNames	long			Offset in table stream of list names string table	Word 97
886	0x0376	lcbSttbListNames	ulong			Count of bytes for the above data	Word 97
890	0x037A	fcSttbfUssr	long			Offset in table stream of undocumented undo / versioning data	Word 97
894	0x037E	lcbSttbfUssr	ulong			Count of bytes for the above data	Word 97
898	0382	fcPlcfTch	FC			Offset in table stream of table chars This is an internal cache used by Word	Word 2000
902	0386	lcbPlcfTch	ulong			Count of bytes of the above data	Word 2000
906	038A	fcRmdfThreading	FC			Offset in table stream of revision mark data This information is unused	Word 2000
910	038E	lcbRmdfThreading	ulong			Count of bytes for the above data	Word 2000

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
914	0392	fcMid	FC			Offset in table stream of Message ID (if any) This information is unused	Word 2000
918	0396	lcbMid	ulong			Count of bytes for the above data	Word 2000
922	039A	fcSttbRgtplc	FC			Offset in table stream of list gallery data (tplcs) This is internal information used by Word's list user interface	Word 2000
926	039E	lcbSttbRgtplc	ulong			Count of bytes for the above data	Word 2000
930	03A2	fcMsoEnvelope	FC			Offset in table stream of persist the mail envelope This is undocumented email header information saved with the file	Word 2000
934	03A6	lcbMsoEnvelope	ulong			Count of bytes for the above data	Word 2000
938	03AA	fcPlcflad	FC			Offset in table stream of Language Auto Detect results This is internal information used by Word's language detection feature	Word 2000
942	03AE	lcbPlcflad	ulong			Count of bytes for the above data	Word 2000
946	03B2	fcRgdofr	FC			Document File Records (miscellaneous document data) This is undocumented miscellaneous information	Word 2000
950	03B6	lcbRgdofr	ulong			Count of bytes for the above data	Word 2000
954	03BA	fcPlcosl	FC			Offset in table stream of NLCheck grammar option state per language This is internal information used by Word's grammar features	Word 2000
958	03BE	lcbPlcosl	ulong			Count of bytes for the above data	Word 2000
962	03C2	fcPlcfcookieOld	FC			Offset in table stream of NLCheck error handle pre Word10 This is internal information used by Word's grammar features	Word 2000
966	03C6	lcbPlcfcookieOld	Ulong			Count of bytes for the above data	Word 2000
970	03CA	rgpgdbkdOld[3]	FCPGDOLD[3] (see definition of this structure above)			Index into the following three properties This is an internal information cache used by Word	Word 2000

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
970	03CA	fcpgdMotherOld	FCPGDOLD			Main document repagination cache: used internally by Word This is an internal information cache used by Word	Word 2000
986	03DA	fcpgdFtnOld	FCPGDOLD			Footnotes repagination cache: used internally by Word This is an internal information cache used by Word	Word 2000
1002	03EA	fcpgdEdnOld	FCPGDOLD			Endnotes repagination cache: used internally by Word This is an internal information cache used by Word	Word 2000
1018	03FA	fcUnused	FC			Not used	Word 2000
1022	03FE	lcbUnused	Ulong			Not used	Word 2000
1026	0402	fcPlcfpgp	FC			Offset in table stream of Paragraph Group Properties This is undocumented HTML DIV (paragraph group) formatting information	Word 2000
1030	0406	lcbPlcfpgp	Ulong			Count of bytes for the above data	Word 2000
1034	040A	fcPlcfuim	FC			Offset in table stream of UIM property data This is internal information used by language input features in Word	Word 2002
1038	040E	lcbPlcfium	Ulong			Count of bytes for the above data	Word 2002
1042	0412	fcPlfguidUim	FC			Offset in table stream of UIM table of GUIDs This is internal information used by language input features in Word	Word 2002
1046	0416	lcbPlfguidUim	Ulong			Count of bytes for the above data	Word 2002
1050	041A	fcAtrdExtra	FC			Offset in table stream of plex of ATRDPost10 structures	Word 2002
1054	041E	lcbAtrdExtra	Ulong			Count of bytes for the above data	Word 2002
1058	0422	fcPlrsid	FC			Offset in table stream of RSID plex. This is undocumented revision mark information.	Word 2002
1062	0426	lcbPlrsid	Ulong			Count of bytes for the above data	Word 2002

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
1066	042A	fcSttbfBkmkFactoid	FC			Offset in table stream of smart tag bookmark STTB This is undocumented information on the smart tags embedded in the document	Word 2002
1070	042E	lcbSttbfBkmkFactoid	Ulong			Count of bytes for the above data	Word 2002
1074	0432	fcPlcfBkfFactoid	FC			Offset in table stream of smart tag bookmark plc of cpFirsts This is undocumented information on the smart tags embedded in the document	Word 2002
1078	0436	lcbPlcfBkfFactoid	Ulong			Count of bytes for the above data	Word 2002
1082	043A	fcPlcfcookie	FC			Offset in table stream of whether the NLCheck error handle will persist in file This is internal information used by Word's grammar features.	Word 2002
1086	043E	lcbPlcfcookie	Ulong			Count of bytes for the above data	Word 2002
1090	0442	fcPlcfBklFactoid	FC			Offset in table stream of smart tag bookmark plc of cpLims This is undocumented information on the smart tags embedded in the document	Word 2002
1094	0446	lcbPlcfBklFactoid	Ulong			Count of bytes for the above data	Word 2002
1098	044A	fcFactoidData	FC			Offset in table stream of smart tag data This is undocumented information on the smart tags embedded in the document	Word 2002
1102	044E	lcbFactoidData	Ulong			Count of bytes for the above data	Word 2002
1106	0452	fcDocUndo	FC			Offset in table stream of undocumented undo / versioning data This is internal information used by Word's undo/versioning features.	Word 2002
1110	0456	lcbDocUndo	Ulong			Count of bytes for the above data	Word 2002
1114	045A	fcSttbfBkmkFcc	FC			Offset in table stream of fcc bookmark sttb This is internal bookmark information used by Word's styles and formatting feature to keep track of formatting use.	Word 2002

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
1118	045E	lcbSttbfBkmkFcc	Ulong			Count of bytes for the above data	Word 2002
1122	0462	fcPlcfBkfFcc	FC			Offset in table stream of fcc bookmark plc of cpFirsts This is internal bookmark information used by Word's styles and formatting feature to keep track of formatting use.	Word 2002
1126	0466	lcbPlcfBkfFcc	Ulong			Count of bytes for the above data	Word 2002
1130	046A	fcPlcfBklFcc	FC			Offset in table stream of fcc bookmark plc of cpLims This is internal bookmark information used by Word's styles and formatting feature to keep track of formatting use.	Word 2002
1134	046E	lcbPlcfBklFcc	Ulong			Count of bytes for the above data	Word 2002
1138	0472	fcSttbfbkmkBPRepairs	FC			Offset in table stream of file repair bookmark sttb This is internal bookmark information used by Word's styles and formatting feature to keep track of formatting use.	Word 2002
1142	0476	lcbSttbfbkmkBPRepairs	Ulong			Count of bytes for the above data	Word 2002
1146	047A	fcPlcfbkfBPRepairs	FC			Offset in table stream of file repair bookmark plc of cpFirsts This is internal bookmark information used by Word's file repair feature to track repaired document portions.	Word 2002
1150	047E	lcbPlcfbkfBPRepairs	Ulong			Count of bytes for the above data	Word 2002
1154	0482	fcPlcfbklBPRepairs	FC			Offset in table stream of file repair bookmark plc of cpLims This is internal bookmark information used by Word's file repair feature to track repaired document portions.	Word 2002
1158	0486	lcbPlcfbklBPRepairs	Ulong			Count of bytes for the above data	Word 2002

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
1162	048A	fcPmsNew	FC			Offset in table stream of new mail merge state information, needed because old ipfnpmf was not validated. This is undocumented information used by Word's mail merge feature.	Word 2002
1166	048E	lcbPmsNew	Ulong			Count of bytes for the above data	Word 2002
1170	0492	fcODSO	FC			Offset in table stream of IMsoODSO/IMsoMailmerge Information This is undocumented information used by Word's mail merge feature.	Word 2002
1174	0496	lcbODSO	Ulong			Count of bytes for the above data	Word 2002
1178	049A	fcPlcfpmiOldXP	FC			Offset in table stream of Paragraph Mark Information (Old View) for Word 2002. This is an internal information cache used by Word.	Word 2002
1182	049E	lcbPlcfpmiOldXP	Ulong			Count of bytes for the above data .	Word 2002
1186	04A2	fcPlcfpmiNewXP	FC			Offset in table stream of Paragraph Mark Information (New View) for Word 2002. This is an internal information cache used by Word.	Word 2002
1190	04A6	lcbPlcfpmiNewXP	Ulong			Count of bytes for the above data.	Word 2002
1194	04AA	fcPlcfpmiMixedXP	FC			Offset in table stream of Paragraph Mark Information (Mixed View) for Word 2002. This is an internal information cache used by Word.	Word 2002
1198	04AE	lcbPlcfpmiMixedXP	Ulong			Count of bytes for the above data.	Word 2002
1202	04B2	fcEncryptedProps	FC			Offset in table stream of encryption properties This is an internal encrypted version of document properties used by Word.	Word 2002
1206	04B6	lcbEncryptedProps	Ulong			Count of bytes for the above data	Word 2002

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
1210	04BA	fcPlcffactoid	FC			Offset in table stream of background factoid checking state This is internal state information used by Word's smart tag features.	Word 2002
1214	O4BE	lcbPlcffactoid	Ulong			Count of bytes for the above data.	Word 2002
1218	04C2	fcPlcflvcOldXP	FC			Offset in table stream of LVC PLC (Old View) for Word 2002. This is an internal information cache used by Word.	Word 2002
1222	04C6	lcbPlcflvcOldXP	Ulong			Count of bytes for the above data.	Word 2002
1226	04CA	fcPlcflvcNewXP	FC			Offset in table stream of LVC PLC (New View) for Word 2002. This is an internal information cache used by Word.	Word 2002
1230	04CE	lbcPlcflvcNewXP	Ulong			Count of bytes for the above data.	Word 2002
1234	04D2	fcPlcflvcMixedXP	FC			Offset in table stream of LVC PLC (Mixed View) for Word 2002. This is an internal information cache used by Word.	Word 2002
1238	04D6	lcbPlcflvcMixedXP	Ulong			Count of bytes for the above data.	Word 2002
1242	4DA	fcHplxsdr	FC			XML Schema Definition References This is undocumented XML schema information. The recommended way to consume this information is through Word's XML output. The XML format is documented separately and can be found on MSDN.	Word 2003
1246	4DE	lcbHplxsdr;	ulong			Count of bytes for the above data.	Word 2003
1250	4E2	fcSttbfBkmkSdt	FC			SDT bookmark STTB This is undocumented XML bookmark information. The recommended way to consume this information is through Word's XML output. The XML format is documented separately and can be found on MSDN.	Word 2003
1254	4E6	lcbSttbfBkmkSdt	ulong			Count of bytes for the above data.	Word 2003

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
1258	4EA	fcPlcfBkfSdt	FC			SDT bookmark plc of cpFirsts This is undocumented XML bookmark information. The recommended way to consume this information is through Word's XML output. The XML format is documented separately and can be found on MSDN.	Word 2003
1262	4EE	lcbPlcfBkfSdt	ulong			Count of bytes for the above data.	Word 2003
1266	4F2	fcPlcfBklSdt	FC			SDT bookmark plc of cpLims This is undocumented XML bookmark information. The recommended way to consume this information is through Word's XML output. The XML format is documented separately and can be found on MSDN.	Word 2003
1270	4F6	lcbPlcfBklSdt	ulong			Count of bytes for the above data.	Word 2003
1274	4FA	fcCustomXForm	FC			Custom XML Transform on save This is undocumented XML transform information. The recommended way to consume this information is through Word's XML output. The XML format is documented separately and can be found on MSDN.	Word 2003
1278	4FE	lcbCustomXForm	ulong			Count of bytes for the above data.	Word 2003
1282	502	fcSttbfBkmkProt	FC			Range protection bookmark STTB This is undocumented bookmark information used by Word's document protection feature.	Word 2003
1286	506	lcbSttbfBkmkProt	ulong			Count of bytes for the above data.	Word 2003
1290	50A	fcPlcfBkfProt	FC			Range protection bookmark plc of cpFirsts This is undocumented bookmark information used by Word's document protection feature.	Word 2003
1294	50E	lcbPlcfBkfProt	ulong			Count of bytes for the above data.	Word 2003

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
1298	512	fcPlcfBklProt	FC			Range protection bookmark plc of cpLims This is undocumented bookmark information used by Word's document protection feature.	Word 2003
1302	516	lcbPlcfBklProt	ulong			Count of bytes for the above data.	Word 2003
1306	51A	fcSttbProtUser	FC			Range protection user list STTB This is undocumented user information used by Word's document protection feature.	Word 2003
1310	51E	lcbSttbProtUser	ulong			Count of bytes for the above data.	Word 2003
1314	522	fcPlcftpc	FC			Current text paragraph cache This is unused.	Word 2003
1318	526	lcbPlcftpc	ulong			Count of bytes for the above data.	Word 2003
1322	52A	fcPlcfpmiOld	FC			Paragraph Mark Information (Old View) This is an internal information cache used by Word.	Word 2003
1326	52E	lcbPlcfpmiOld	ulong			Count of bytes for the above data.	Word 2003
1330	532	fcPlcfpmiOldInline	FC			Paragraph Mark Information (Old Inline View) This is an internal information cache used by Word.	Word 2003
1334	536	lcbPlcfpmiOldInline	ulong			Count of bytes for the above data.	Word 2003
1338	53A	fcPlcfpmiNew	FC			Paragraph Mark Information (New View) This is an internal information cache used by Word.	Word 2003
1342	53E	lcbPlcfpmiNew	ulong			Count of bytes for the above data.	Word 2003
1346	542	fcPlcfpmiNewInline	FC			Paragraph Mark Information (New Inline View) This is an internal information cache used by Word.	Word 2003
1350	546	lcbPlcfpmiNewInline	ulong			Count of bytes for the above data.	Word 2003
1354	54A	fcPlcflvcOld	FC			LVC PLC (Old View) This is an internal information cache used by Word.	Word 2003
1358	54E	lcbPlcflvcOld	ulong			Count of bytes for the above data.	Word 2003
1362	552	fcPlcflvcOldInline;	FC			LVC PLC (Old Inline View) This is an internal information cache used by Word.	Word 2003

Decimal	Hex	Name	Type	Bitfield Size	Bitfield Mask	Comments	Introduced
1366	556	lcbPlcflvcOldInline	ulong			Count of bytes for the above data.	Word 2003
1370	55A	fcPlcflvcNew	FC			LVC PLC (New View) This is an internal information cache used by Word.	Word 2003
1374	55E	lcbPlcflvcNew	ulong			Count of bytes for the above data.	Word 2003
1378	562	fcPlcflvcNewInline	FC			LVC PLC (New Inline View) This is an internal information cache used by Word.	Word 2003
1382	566	lcbPlcflvcNewInline	ulong			Count of bytes for the above data.	Word 2003
1386	56A	rgpgdbkd[3]	FCPGD			This is an internal information cache used by Word.	Word 2003
1386	56A	fcpgdMother	FCPGD			This is an internal information cache used by Word.	Word 2003
1410	582	fcpgdFtn	FCPGD			This is an internal information cache used by Word.	Word 2003
1434	59A	fcpgdEdn	FCPGD			This is an internal information cache used by Word.	Word 2003
1458	5B2	fcAfd	FC			This is internal revision mark view information used by Word.	Word 2003
1462	5B6	lcbAfd	ulong			Count of bytes for the above data.	Word 2003
1466	5BA	cswNew	Ushort			The number of entries in rgswNew[]	Word 2003
1468	5Bc	rgswNew[]	Ushort			Index to the following properties	Word 2003
1468	5Bc	nFib	Ushort			The actual nFib, moved here because some readers assumed they couldn't read any format with nFib > some constant	Word 2003
1470	5BE	cQuickSavesNew	Ushort			Because of the above, we need to use cQuickSaves to prevent Word 97 from quick saving to Word 2000 files	Word 2003

[1] **Note**: when ccpFtn==0 and ccpHdr==0 and ccpMcr==0 and ccpAtn==0 and ccpEdn==0 and ccpTxbx==0 and ccpHdrTxbx==0, then fib.fcMac=fib.fcMin+ fib.ccpText. If either ccpFtn!=0 or ccpHdd!=0 or ccpMcr!=0 or ccpAtn!=0 or ccpEdn!=0 or ccpTxbx!=0 or ccpHdrTxbx==0, then fib.fcMac=fib.fcMin+fib.ccpText+fib.ccpFtn+fib.ccpHdd+fib.ccpMcr+fib.ccpAtn+fib.ccpEdn+ fib.ccpTxbx+fib.ccpHdrTxbx+1. The single character stored beginning at file position fib.fcMac-1 must always be a CR character (ASCII 13).

cbFIB (count of bytes of FIB) is 1472 (decimal), 5C0 (hex).

Note If a table does not exist in the file, its cb in the FIB is zero and its fc is equal to that of the following table (the latter equality is irrelevant, as the cb should be used to determine existence of the table).

Footnote Reference Descriptor (FRD)

The FRD is stored in both the plcffndRef and the plcfendRef.

Offset (base 10)	Field	Type	Size	Bitfield	Comments
0		nAuto	short		If > 0, the note is an automatically numbered note, otherwise it has a custom mark

Formatted Disk Page for CHPXs (CHPX FKP)

Offset (base 10)	Field	Type	Size	Bitfield	Comments
0		Rgfc	array of FCs		Each FC is the limit FC of a run of exception text
4*(fkp.crun+1)	rgb	array of bytes			An array of bytes where each byte is the word offset of a CHPX. If the byte stored is 0, there is no difference between run's character properties and the style's character properties.
5*fkp.crun+4		unused space			As new runs/paragraphs are recorded in the FKP, unused space is reduced by 5 if CHPX is already recorded and is reduced by 5+sizeof(CHPX) if property is not already recorded.
511-sizeof(grpchpx)	grpchpx	array of bytes			Grpchpx consists of all of the CHPXs stored in FKP concatenated end to end. Each CHPX is prefixed with a count of bytes which records its length.
511		Crun	byte		Count of runs for CHPX FKP

The CHP is never stored in a Word file. It is derived by expanding stored CHPXs.

Formatted Disk Page for PAPXs (PAPX FKP)

Offset (base 10)	Field	Type	Size	Bitfield	Comments
0	rgfc	FC[fkp.crun+1]			Each FC is the limit FC of a paragraph (i.e. points to the next character past an end of paragraph mark). There will be fkp.crun+1 recorded in the FKP.
4*(fkp.crun+1)	rgbx	BX[fkp.crun]			An array of the BX data structure. The ith BX entry in the array describes the paragraph beginning at fkp.rgfc[i]. The BX is a 13 byte data structure. The first byte of each BX is the word offset of the PAPX recorded for the paragraph corresponding to this BX. If the byte stored is 0, this represents a 1 line paragraph 15 pixels high with Normal style (stc == 0) whose column width is 7980 dxas. The last 12 bytes of the BX is a PHE structure which stores the current paragraph height for the paragraph corresponding to the BX. If a plcfphe has an entry that maps to the FC for this paragraph, that entry's PHE overrides the PHE stored in the FKP.11*fkp.crun+4 unused space. As new runs/paragraphs are recorded in the FKP, unused space is reduced by 17 if CHPX/PAPX is already recorded and is reduced by 17+sizeof(PAPX) if property is not already recorded.
511-size of(grppapx)	grppapx	array of bytes			grppapx consists of all of the PAPXs stored in FKP concatenated end to end. Each PAPX begins with a count of words which records its length padded to a word boundary.
511	crun	Byte			Count of paragraphs for PAPX FKP.

The PAP is never stored in a Word file. It is derived by expanding stored PAPXs.

Hyphenation (HRESI)

Substructure of the CHP. Referenced elsewhere in this document.

b10	b16	Field	Type	Size	Bitfield	Comment
0	0	hres	Uns char	1		Hyphenation rule 0 No hyphenation 1 Normal hyphenation 2 Add letter before hyphen 3 Change letter before hyphen 4 Delete letter before hyphen 5 Change letter after hyphen 6 Delete letter before the hyphen and change the letter preceding the deleted character
1	1	chHres	Uns char	1		The character that will be used to add or change a letter when chp.ysr is 2,3, 5 or 6

List LeVeL (on File) (LVLF)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0x00	iStartAt	Long	4		Start at value for this list level
4	0x04	nfc	Byte	1		Number format code (see anld.nfc for a list of options)
5	0x05	jc	uns char	:2	0x03	Alignment (left, right, or centered) of the paragraph number.
		fLegal	uns char	:1	0x04	True (==1) if the level turns all inherited numbers to arabic, false if it preserves their number format code (nfc)
		fNoRestart	uns char	:1	0x08	True if the level's number sequence is not restarted by higher (more significant) levels in the list
		fPrev	uns char	:1	0x10	Word 6.0 compatibility option: equivalent to anld.fPrev (see ANLD)
		fPrevSpace	uns char	:1	0x20	Word 6.0 compatibility option: equivalent to anld.fPrevSpace (see ANLD)
		fWord6	uns char	:1	0x40	True if this level was from a converted Word 6.0 document. If it is true, all of the Word 6.0 compatibility options become valid; otherwise they are ignored.
6	0x06	rgbxchNums[9]	Array	9		Contains the character offsets into the LVL's XST of the inherited numbers of previous levels. This array should be zero terminated unless it is full (all 9 levels full). The XST contains place holders for any paragraph numbers contained in the text of the number, and the place holder contains the ilvl of the inherited number, so lvl.xst[lvl.rgbxchNums[0]] == the level of the first inherited number in this level.
15	0x0F	ixchFollow	uns char	1		The type of character following the number text for the paragraph: 0 == tab, 1 == space, 2 == nothing.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
16	0x10	dxaSpace	Long	4		Word 6.0 compatibility option: equivalent to anld.dxaSpace (see ANLD). For newer versions indent to remove if we remove this numbering.
20	0x14	dxaIndent	Long	4		Word 6.0 compatibility option: equivalent to anld.dxaIndent (see ANLD). Unused in newer versions.
24	0x18	cbGrpprlChpx	Byte	1		Length, in bytes, of the LVL's grpprlChpx
25	0x19	cbGrpprlPapx	Byte	1		Length, in bytes, of the LVL's grpprlPapx
26	0x1A	ilvlRestartLim	Uchar	1		Limit of levels that we restart after
27	0x1B	grfhic	Uchar	1		HTML compatibility flags: 0x01 Checked 0x02 The numbering sequence or format is unsupported (includes tab & size) 0x04 The list text is not "#." 0x080 Something other than a period is used 0x10 First line indent mismatch 0x20 The list tab and the dxaLeft don't match (need table?) 0x40 The hanging indent falls beneath the number (need plain text) 0x80 A built-in HTML bullet

Line Spacing Descriptor (LSPD)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	dyaLine	short			See description of sprmPDyaLine for description of the meaning of dyaLine
2	2	fMultLinespace	short			See description of sprmPDyaLine in the Sprm Definitions section for description of the meaning of dyaLine and fMultLinespace fields

cbLSPD (count of bytes of LSPD) is 4.

LiST Data (on File) (LSTF)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0x00	lsid	long	4		Unique List ID
4	0x04	tplc	long	4		Unique template code
8	0x08	rgistd[9]	array	18		Array of shorts containing the istd's linked to each level of the list, or istdNil (4095) if no style is linked.
26	0x1A	fSimpleList	uns char	:1	0x01	True if this is a simple (one-level) list; false if this is a multilevel (nine-level) list.
		fRestartHdn	uns char	:1	0x02	Word 6.0 compatibility option: true if the list should start numbering over at the beginning of each section
		fAutoNum	uns char	:1	0x04	To emulate Word 6.0 numbering: true if Auto numbering

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
		fPreRTF	uns char	:1	0x08	When 1, this list was there before we started reading RTF
		fHybrid	uns char	:1	0x10	When 1, list is a hybrid multilevel/simple (UI=simple, internal=multilevel)
		reserved	uns char	:3	0xE0	Reserved
27	0x1B	grfhic	uns char			HTML compatibility flags: 0x01 Checked 0x02 The numbering sequence or format is unsupported (includes tab & size) 0x04 The list text is not "#." 0x080 Something other than a period is used 0x10 First line indent mismatch 0x20 The list tab and the dxaLeft don't match (need table?) 0x40 The hanging indent falls beneath the number (need plain text) 0x80 A built-in HTML bullet

List Format Override (LFO)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0x0	lsid	long	4		List ID of corresponding LSTF (see LSTF)
4	0x4	reserved	long	4		Reserved
8	0x8	reserved	long	4		Reserved
12	0xC	clfolvl	uns char	1		Count of levels whose format is overridden (see LFOLVL)
13	0xD	ibstFltAutoNum	uns char	1		Used for AUTONUM field emulation
14		grfhic	uns char	1		HTML compatibility flags: 0x01 Checked 0x02 The numbering sequence or format is unsupported (includes tab & size) 0x04 The list text is not "#." 0x080 Something other than a period is used 0x10 First line indent mismatch 0x20 The list tab and the dxaLeft don't match (need table?) 0x40 The hanging indent falls beneath the number (need plain text) 0x80 A built-in HTML bullet
15		reserved	uns char	1		Reserved

List Format Override for a single LeVeL (LFOLVL)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	iStartAt	long	4		Start-at value if fFormatting==false and fStartAt==true. (if fFormatting == true, the start- is stored in the LVL)
4	4	ilvl	uns long	:4	0x0F	The level to be overridden
		fStartAt	uns long	:1	0x10	True if the start-at value is overridden
		fFormatting	uns long	:1	0x20	True if the formatting is overridden (in which case the LFOLVL should contain a pointer to a LVL)
		grfhic	uns long	:8	0x3FC0	HTML compatibility flags: 0x01 Checked 0x02 The numbering sequence or format is unsupported (includes tab & size) 0x04 The list text is not "#." 0x080 Something other than a period is used 0x10 First line indent mismatch 0x20 The list tab and the dxaLeft don't match (need table?) 0x40 The hanging indent falls beneath the number (need plain text) 0x80 A built-in HTML bullet
		reserved	uns long	:18		Reserved

Outline LiST Data (OLST)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	rganlv[9]	ANLV			An array of 9 ANLV structures describing how heading numbers should be displayed for each of Word's 9 outline heading levels
180	B4	fRestartHdr	uns char			When ==1, restart heading on section break
181	B5	fSpareOlst2	uns char			Reserved
182	B6	fSpareOlst3	uns char			Reserved
183	B7	fSpareOlst4	uns char			Reserved
184	B8	rgxch[32]	array of 32 XCHARs			Text before/after number

cbOLST (count of bytes of OLST) is 248(decimal), F8(hex).

Outline LiST Data for Word 97 (OLST80)

Same as OLST but rganlv[9] is of type ANLV80.

Number Revision Mark Data (NUMRM)

The NUMRM structure is used to track revision marking data for paragraph numbers, and is stored in the PAP for each numbered paragraph. When revision marking tracking is turned on, we fill out the NUMRM for each number with the data required to recreate the number's text. Then at display time, that string is compared with the current paragraph number string, and displayed as changed (old deleted, current inserted) if the strings differ. The string construction algorithm is the same as for an LVL structure.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
0	0	fNumRM	uns char	1		True if this paragraph was numbered when revision mark tracking was turned on
1	1	Spare	uns char	1		
2	2	ibstNumRM	short	2		Index to author IDs stored in hsttbfRMark for the paragraph number change
4	4	dttmNumRM	DTTM	4		Date of the paragraph number change
8	8	rgbxchNums[9]	uns char[9]	9		Index into NUMRM.xst of the locations of paragraph number place holders for each level (see LVL.rgxchNums)
17	11	rgnfc[9]	uns char[9]	9		Number format code for the paragraph number place holders for each level (see LVL.nfc)
26	1A	Spare	short	2		
28	1C	PNBR	int [9]	36		Numeric value for each level place holder in NUMRM.xst.
64	40	xst	XCHAR[32]	64		The text string for the paragraph number, containing level place holders

cbNUMRM (count of bytes of NUMRM) is 128 (decimal), 80 (hex).

Page Descriptor (PGD)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	*	short	:4	000F	
		fGhost	short	:2	0030	Redefine fEmptyPage and fAllFtn. true when blank page or footnote only page
		*	short	:10	FFC0	
0	0	fContinue	short	:1	0001	When 1, footnote is continued from previous page
		fUnk	short	:1	0002	When 1, page is dirty (i.e. pagination cannot be trusted)
		fRight	short	:1	0004	When 1, right hand side page
		fPgnRestart	short	:1	0008	When 1, page number must be reset to 1
		fEmptyPage	short	:1	0010	When 1, section break forced page to be empty.
		fAllFtn	short	:1	0020	When 1, page contains nothing but footnotes
			short	:1	0040	Unused
		fTableBreaks	short	:1	0080	Table breaks have been calculated for this page
		fMarked	short	:1	0100	Used temporarily while word is running
		fColumnBreaks	short	:1	0200	Column breaks have been calculated for this page
		fTableHeader	short	:1	0400	Page had a table header at the end

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
		fNewPage	short	:1	0800	Page has never been valid since created, must recalculate the bounds of this page. If this is the last page, this PGD may really represent many pages
		bkc	short	:4	F000	Section break code
2	2	lnn	uns short			Line number of first line, -1 if no line numbering
4	4	pgn	uns short			Page number as printed
6	6	dym	long			Height of page for online view
10	A	dxm	long			Width of page for web view
14	E	dyaPage	long			Page height for web or page view

cbPGD (count of bytes of PGD) is 18.

Paragraph Height (PHE)

The PHE is a substructure of the PAP and the PAPX FKP and is also stored in the PLCFPHE.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	fVolatile	short	:1	0001	Complex shape layout in this paragraph
		fUnk	short	:1	0002	PHE entry is invalid when == 1
		fDiffLines	short	:1	0004	When 1, total height of paragraph is known but lines in paragraph have different heights
		*	short	:5	00F8	Reserved
		clMac	short	:8	FF00	When fDiffLines is 0 is number of lines in paragraph
2	2		short			Reserved
4	4	dxaCol	long			Width of lines in paragraph
8	8	dymLine	long			When fDiffLines is 0, is height of every line in paragraph in pixels
8	8	dymHeight	long			When fDiffLines is 1, is the total height in pixels of the paragraph

If the PHE is stored in a PAP whose fTtp field is set (non-zero), the following structure is used:

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	fSpare	short	:1	0001	Reserved
		fUnk	short	:1	0002	PHE entry is invalid when == 1
		dcpTtpNext	short	:30		If not == 0, used as a hint when finding the next row
4	4	dxaCol	long			
8	8	dymTableHeight	long			Height of table row

cbPHE (the count of bytes of PHE) is 12.

If there is no paragraph height information stored for a paragraph, all of the fields in the PHE are set to 0. If a paragraph contains more than 127 lines, the clMac, dylLine variant cannot be used, so fDiffLines must be set to 1 and the total size of the paragraph stored in dylHeight. If a paragraph height is greater than 32767 twips, the height cannot be represented by a PHE so all fields of the PHE must be set to 0.

If a new Word file is created, the PHE of every papx fkp entry created to describe the paragraphs of the file should be set to 0. If a Word file is altered in place (a character of the file changed to a new character or a property changed), the paragraph containing the change must have its papx.phe field set to 0. If this paragraph is in a table row, the PHE in the papx at the end of the row (indicated by fInTable) must also be set to 0.

Paragraph Properties (PAP)

The PAP is never written to a file. Word only writes the SPRMs that modify their respective properties. From them one can build a the collection of paragraph properties. For this reason offsets into the structure are not listed

Field	Type	Size	Comments
istd	ushort	2	Index to style descriptor. This is an index to an STD in the STSH structure.
fSideBySide	uchar	1	When 1, paragraph is a side by side paragraph
spare1	uchar	1	Unused
fKeep	uchar	1	Keep entire paragraph on one page if possible
fKeepFollow	uchar	1	Keep paragraph on same page with next paragraph if possible
fPageBreakBefore	uchar	1	Start this paragraph on new page
brcl	uchar	1	Border line style 0 single 1 thick 2 double 3 shadow
brcp	uchar	1	Rectangle border codes 0 none 1 border above 2 border below 15 box around 16 bar to left of paragraph
ilvl	uchar	1	When non-zero, list level for this paragraph
ilfo	uchar	1	When non-zero, (1-based) index into the pllfo identifying the list to which the paragraph belongs
fNoLnn	uchar	1	No line numbering for this paragraph. (makes this an exception to the section property of line numbering)
nLvlAnm	uchar	1	Unused
lspd	LSPD	4	Line spacing descriptor for the paragraph
dyaBefore	Unsigned long	4	Space before paragraph (unsigned)
dyaAfter	Unsigned long	4	Space after paragraph (unsigned)

Field	Type	Size	Comments
finTableW97	uchar	1	Paragraph is in table (archaic)
fTtp	uchar	1	Table trailer paragraph (last paragraph in table row)
swUnused2	short	2	Unused
dxaAbs	long	4	When positive, is the horizontal distance from the reference frame specified by pap.pcHorz. 0 means paragraph is positioned at the left with respect to the reference frame specified by pcHorz. Certain negative values have special meaning: -4 paragraph centered horizontally within reference frame -8 paragraph adjusted right within reference frame -12 paragraph placed immediately inside of reference frame -16 paragraph placed immediately outside of reference frame
dyaAbs	long	4	When positive, is the vertical distance from the reference frame specified by pap.pcVert. 0 means paragraph's y-position is unconstrained. Certain negative values have special meaning: -4 paragraph is placed at top of reference frame -8 paragraph is centered vertically within reference frame -12 paragraph is placed at bottom of reference frame
dxaWidth	long	4	When not == 0, paragraph is constrained to be dxaWidth wide, independent of current margin or column settings
fBrLnAbove	uns char	:1	
fBrLnBelow	uns char	:1	
fUnused	uns char	:2	Unused
pcVert	uns char	:2	Vertical position code. Specifies coordinate frame to use when paragraphs are absolutely positioned. 0 vertical position coordinates are relative to margin 1 coordinates are relative to page 2 coordinates are relative to text. This means: relative to where the next non-APO text would have been placed if this APO did not exist.
pchorz	uns char	:2	Horizontal position code. Specifies coordinate frame to use when paragraphs are absolutely positioned. 0 horizontal position coordinates are relative to column. 1 coordinates are relative to margin 2 coordinates are relative to page
unused	uns char	:8	Unused
wr	BYTE	1	Wrap code for absolute objects
fNoAutoHyph	uchar	1	When 0, text in paragraph may be auto hyphenated
wHeightAbs	ushort	2	Height when 0 == Auto
dyaHeight		:15	Height of abs obj; 0 == Auto
fMinHeight		:1	Minimum height is exact or auto: 0 = Exact 1 = At Least

Field	Type	Size	Comments
dcs	DCS	2	Drop cap specifier
dyaFromText	long	4	Vertical distance between text and absolutely positioned object
dxaFromText	long	4	Horizontal distance between text and absolutely positioned object
fLocked	BYTE	1	Anchor of an absolutely positioned frame is locked
fWidowControl	uchar	1	When 1, Word will prevent widowed lines in this paragraph from being placed at the beginning of a page
fKinsoku	uns char	1	When 1, apply Kinsoku rules when performing line wrapping
fWordWrap	uns char	1	When 1, perform word wrap
fOverflowPunct	uns char	1	When 1, apply overflow punctuation rules when performing line wrapping
fTopLinePunct	uns char	1	When 1, perform top line punctuation processing
fAutoSpaceDE	uns char	1	When 1, auto space East Asian and alphabetic characters
fAtuoSpaceDN	uns char	1	When 1, auto space East Asian and numeric characters
wAlignFont	uns short	2	Font alignment 0 Hanging 1 Centered 2 Roman 3 Variable 4 Auto
fVertical	short	:1	Used internally by Word
fBackward	short	:1	Used internally by Word
fRotateFont	short	:1	Used internally by Word
empty	short	:13	Reserved
iSnapBaseLine	uchar	1	Unused
lvl	char	1	Outline level
fBiDi	uchar	1	Text flows right to left.
fNumRMins	uchar	1	Paragraph number is inserted (only valid if pap.numrm.fNumRM is 0)
fCrLf	uchar	1	Used internally
fUsePgsuSettings	uchar	1	Use Page Setup Line Pitch
fAdjustRight	uchar	1	Adjust right margin
fUnused2	uchar	1	Unused
itap	long	4	Table nesting level
fInnerTableCell	uchar	1	When 1, the end of paragraph mark is really an end of cell mark for a nested table cell

Field	Type	Size	Comments
fOpenTch	uchar	1	Ensure the Table Cell char doesn't show up as zero height
dxcRight	short	2	Right indent in character units
dxcLeft	short	2	Left indent in character units
dxcLeft1	short	2	First line indent in character units
dylBefore	short	2	Vertical spacing before paragraph in character units
dylAfter	short	2	Vertical spacing after paragraph in character units
fDyaBeforeAuto	uchar	1	Vertical spacing before is automatic
fDyaAfterAuto	uchar	1	Vertical spacing after is automatic
dxaRight	long	4	Word 97: indent from right margin Word 2000: indent from right margin (signed) for left-to-right text; from left margin for right-to-left text
dxaLeft	long	4	Word 97: indent from left margin (signed) Word 2000: indent from left margin (signed) for left-to-right text; from right margin for right-to-left text
dxaLeft1	long	4	First line indent; signed number relative to dxaLeft
jc	uchar	1	Justification code 0 left justify 1 center 2 right justify 3 left and right justify 4 distributed 5 Medium 6 List tab 7 High 8 Low 9 Thai distributed Justification in Word 2000 and above is relative to text direction (for example, left is left for left-to-right text and right for right-to-left text).
fNoAllowOverlap	uchar	1	When 1, absolutely positioned paragraph cannot overlap with another paragraph
rgbrc[cbrcParaBorders]	BRC[cbrcParaBorders]	48	Array of borders
brcTop	BRC	8	Specification for border above paragraph
brcLeft	BRC	8	Specification for border to the left of paragraph
brcBottom	BRC	8	Specification for border below paragraph
brcRight	BRC	8	Specification for border to the right of paragraph
brcBetween	BRC	8	Specification of border to place between conforming paragraphs. Two paragraphs conform when both have borders, their brcLeft and brcRight matches, their widths are the same, they both belong to tables or both do not, and have the same absolute positioning props.
brcBar	BRC	8	Specification of border to place on outside of text when facing pages are to be displayed

Field	Type	Size	Comments
shd	SHD	10	Paragraph shading
anld	ANLD	88	Word 6.0 paragraph numbering
phe	PHE	12	Height of current paragraph.
fPropRMark	Uns short	2	When 1, properties have been changed with revision marking on
ibstPropRMark	IBST	2	Index to author IDs stored in hsttbfRMark. Used when properties have been changed when revision marking was enabled
dttmPropRMark	DTTM	4	Date/time at which properties of this were changed for this run of text by the author. (Only recorded when revision marking is on.)
fCharLineUnits	uchar	1	Used internally by Word
fFrpTap	uchar	1	Used internally by Word
dxaFromTextRight	long	4	Used internally by Word
dyaFromTextBottom	long	4	Used internally by Word
lfrp	long	4	Used internally by Word
itbdMac	short	2	Number of tabs stops defined for paragraph. Must be >= 0 and <= 64.
rgdxaTab[itbdMax]	short[itbdMax]	128	Array of positions of itbdMac tab stops. itbdMax==64
rgtbd[itbdMax]	TBD[itbdMax]	64	Array of itbdMac tab descriptors
numrm	NUMRM	128	Paragraph numbering revision mark data (see NUMRM)
ptap	Pointer to a TAP	4	Used internally by Word

The following properties were added in Word 2002:

fNoAllowOverlap	uchar	1	When 1, absolutely positioned paragraph cannot overlap with another paragraph
ipgp	ulong	4	HTML DIV ID for this paragraph
rsid	RSID (long)	4	Save ID for last time this PAP was revised Random number associated with paragraph formatting which improves the accuracy of Word's document merge feature
istdList	ushort	2	Paragraph List Style
fContextualSpacing	uchar	1	Ignore the space before/after properties between paragraphs of the same style
fHasOldProps	uchar	1	Used for paragraph property revision marking. The pap at the time fHasOldProps is set to 1, the is the old pap.
rpf	RPF	1	revision pane flags
hplcnf	HPL	4	Conditional paragraph properties

yfti	YFTI	13	information about the last table autofit conditional results

The standard PAP is all zeros except:

fWidowControl	1
fMultLineSpace	1
dyaLine	240 twips
Lvl	9

Paragraph Property Exceptions (PAPX)

The PAPX is stored within FKPs and within the STSH.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	cb	byte			Count of bytes of following data in PAPX. The first byte of a PAPX is a count of bytes when a PAPX is stored in a STSH. Count of bytes is used because only paragraph sprms are stored in a STSH PAPX.
0	0	cw	byte			Count of words for this byte and the following data in PAPX. The first byte of a PAPX is a count of words when PAPX is stored in an FKP. If this value is 0, it is a 'pad' byte and the count is stored in the following byte, Count of words is used because PAPX in an FKP can contain paragraph and table sprms.
1	1	(cw)	byte			If previous byte is 0, this is the count of words of following data in PAPX (not including this and previous 'pad' byte)
1/2	1/2	istd	uns short			Index to style descriptor of the style from which the paragraph inherits its paragraph and character properties
3/4	3/4	grpprl	character array			A list of the sprms that encode the differences between PAP for a paragraph and the PAP for the style used. When a paragraph bound is also the end of a table row, the PAPX also contains a list of table sprms which express the difference of table row's TAP from an empty TAP that has been cleared to zeros. The table sprms are recorded in the list after all of the paragraph sprms. See sprms definitions for list of sprms that are used in PAPXs.

For calculating papx.cw when storing in an FKP: For even-sized grpprls, the grpprl plus the istd and cw bytes will be an even number of bytes, so we store the count of words for all three elements in papx.cw. For odd-sized grpprls, the three elements will be an odd number of bytes, which can't be represented with a count of words; so, we store a 'pad' byte of 0 at the beginning (in the normal cw location), followed by a count that is the size of the grpprl and istd byte only (since that's an even number of bytes). In either case, papx.cw is immediately followed by the istd and grpprl.

Picture Bullet Information (PBI)

Used within the CHP structure.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	fPicBullet	unsigned short	:1		It is a picture bullet

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
		fNoAutoSize	unsigned short	:1		Do not auto size the picture bullet
		fDefaultPic	unsigned short	:1		This is the default picture bullet
		fTemporary	unsigned short	:1		
		unused	unsigned short	:11		Not used
		fFormatting	unsigned short	:1		Not used. Always set to 0.
2	2	iBullet	CP	4		Character position

Picture Descriptor (on File) (PICF)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	lcb	long			Number of bytes in the PIC structure plus size of following picture data which may be a Window's metafile, a bitmap, or the filename of a TIFF file. In the case of a Macintosh PICT picture, this includes the size of the PIC, the standard "x" metafile, and the Macintosh PICT data. See Appendix B for more information.
4	4	cbHeader	unsigned			Number of bytes in the PIC (to allow for future expansion).
6	6	mfp.mm	short			
8	8	mfp.xExt	short			
10	A	mfp.yExt	short			
12	C	mfp.hMF	short			

If a Windows metafile is stored immediately following the PIC structure, the mfp is a Window's METAFILEPICT structure. See http://msdn2.microsoft.com/en-us/library/ms649017(VS.85).aspx for more information about the METAFILEPICT structure and http://download.microsoft.com/download/0/B/E/0BE8BDD7-E5E8-422A-ABFD-4342ED7AD886/WindowsMetafileFormat(wmf)Specification.pdf for Windows Metafile Format specification.

When the data immediately following the PIC is a TIFF filename, mfp.mm==98 If a bitmap is stored after the pic, mfp.mm==99.

When the PIC describes a bitmap, mfp.xExt is the width of the bitmap in pixels and mfp.yExt is the height of the bitmap in pixels.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
14	E	bm	BITMAP(14 bytes)			Window's bitmap structure when PIC describes a BITMAP
14	E	rcWinMF	rc (rectangle - 8 bytes)			Rectangle for window origin and extents when metafile is stored -- ignored if 0
28	1C	dxaGoal	short			Horizontal measurement in twips of the rectangle the picture should be imaged within

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
30	1E	dyaGoal	short			Vertical measurement in twips of the rectangle the picture should be imaged within

when scaling bitmaps, dxaGoal and dyaGoal may be ignored if the operation would cause the bitmap to shrink or grow by a non -power-of-two factor.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
32	20	mx	uns short			Horizontal scaling factor supplied by user expressed in .001% units
34	22	my	uns short			Vertical scaling factor supplied by user expressed in .001% units

For all of the Crop values, a positive measurement means the specified border was moved inward from its original setting and a negative measurement means the border was moved outward from its original setting.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
36	24	dxaCropLeft	short			The amount the picture has been cropped on the left in twips
38	26	dyaCropTop	short			The amount the picture has been cropped on the top in twips
40	28	dxaCropRight	short			The amount the picture has been cropped on the right in twips
42	2A	dyaCropBottom	short			The amount the picture has been cropped on the bottom in twips
44	2C	brcl	short	:4	000F	Obsolete, superseded by brcTop, etc. In WinWord 1.x, it was the type of border to place around picture 0 single 1 thick 2 double 3 shadow
		fFrameEmpty	short	:1	0010	Picture consists of a single frame
		fBitmap	short	:1	0020	==1, when picture is just a bitmap
		fDrawHatch	short	:1	0040	==1, when picture is an active OLE object
		fError	short	:1	0080	==1, when picture is just an error message
		bpp	short	:8		Bits per pixel 0 unknown 1 monochrome 4 VGA
46	2E	brcTop	BRC			Specification for border above picture
54	36	brcLeft	BRC			Specification for border to the left of picture
62	3E	brcBottom	BRC			Specification for border below picture
70	46	brcRight	BRC			Specification for border to the right of picture
78	4E	dxaOrigin	short			Horizontal offset of hand annotation origin

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
80	50	dyaOrigin	short			Vertical offset of hand annotation origin
82	52	cProps	short			Unused
84	54	rgb				Variable array of bytes containing Window's metafile, bitmap or TIFF file filename

Piece Descriptor (PCD)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
0	0	fNoParaLast	Uns short	:1	0001	When 1, means that piece contains no end of paragraph marks
		fPaphNil	Uns short	:1	0002	Used internally by Word
		fDirty	Uns short	:1	0004	Used internally by Word
		*	Uns short	:1		
1	1	fn	Uns short	:12	FF00	Used internally by Word
2	2	fc	FC			File offset of beginning of piece. The size of the **ith** piece can be determined by subtracting rgcp[i] of the containing plcfpcd from its rgcp[i+1].
6	6	prm	PRM			Contains either a single sprm or else an index number of the grpprl which contains the sprms that modify the properties of the piece

cbPCD (count of bytes of PCD) is 8.

Plex of CPs stored in File (PLCF)

Offset (in decimal)	Field	Type	Comment
0	rgfc	FC[]	The size of PLCF is cb and the size of the structure stored in plc is cbStruct, then the number of structure instances stored in PLCF, iMac is given by (cb -4)/(4 + cbStruct). The number of FCs stored in the PLCF will be iMac + 1.
4*(iMac+1)	rgstruct	struct[]	Array of some arbitrary structure

cbPLC (count of bytes of a PLC) is iMac(4+cbStruct)+4.

Property Modifier(variant 1) (PRM)

The PRM has two variants. In the first variant, the PRM records a single sprm with a bit- or byte-sized operand.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
0	0	fComplex	short	:1	0001	Set to 0 for variant 1
		isprm	short	:7	00FE	Index to entry into rgsprmPrm
		val	short	:8	FF00	sprm's operand

cbPRM (count of bytes of PRM) is 2.

Property Modifier(variant 2) (PRM)

In the second variant, prm.fComplex is 1, and the rest of the structure records an index to a grpprl stored in the CLX (described in **Complex File Format** topic).

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
0	0	fComplex	short	:1	0001	Set to 1 for variant 2
		igrpprl	short	:15	FFFE	Index to a grpprl stored in CLX portion of file

cbPRM (count of bytes of PRM) is 2.

Routing Slip (RS)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	fRouted	short			When 1, document has been routed to at least one recipient
2	2	fReturnOrig	short			When 1, document should be routed to the originator after it has been routed to all recipients
4	4	fTrackStatus	short			When 1, a status message is sent to the originator each time the document is forwarded to a recipient on the routing list
6	6	fDirty	short			Unused(should be 0)
8	8	nProtect	short			Document protection while routing: 0 recipients can make changes to the document and all changes are untracked. 1 recipients can add annotations and make changes to the document. Any changes are tracked by revision marks, and revision marking cannot be turned off. 2 recipients can only add annotations to the document. 3 recipients can enter information only in form fields.
10	A	iStage	short			Index of the current recipient
12	C	delOption	short			When 0, document is routed to each recipient in turn. when 1, document is routed to all recipients simultaneously
14	E	cRecip	short			Count of recipients

cbRS (count of bytes of RS) is 16 (decimal), 10 (hex).

Routing Recipient (RR)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	cb	short			Count of bytes of private system data
2	2	cbSzRecip	short			Count of bytes in recipient string (including null terminator)

cbRR (count of bytes of RR) is 4.

Section Descriptor (SED)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	fn	short			Used internally by Word
2	2	fcSepx	FC			File offset in main stream to beginning of SEPX stored for section. If sed.fcSepx==0xFFFFFFFF, the section properties for the section are equal to the standard SEP (see SEP definition).
6	6	fnMpr	short			Used internally by Word
8	8	fcMpr	FC			Points to offset in FC space of main stream where the Macintosh Print Record for a document created on a Macintosh will be stored

cbSED (count of bytes of SED) is 12 (decimal), C (hex).

Section Properties (SEP)

Note: offsets are not provided for this structure because it is not written to a file. Only Section SPRMs from the collection of section properties can be constructed with total flexibility.

Field	Type	Size	Comments
bkc	uns char		Break code: 0 No break 1 New column 2 New page 3 Even page 4 Odd page
fTitlePage	uns char		Set to 1 when a title page is to be displayed
fAutoPgn	char		Only for Macintosh compatibility, used only during open, when 1, sep.dxaPgn and sep.dyaPgn are valid page number locations
nfcPgn	uns char		Page number format code: 0 Arabic 1 Roman (upper case) 2 Roman (lower case) 3 Letter (upper case) 4 Letter (lower case)
fUnlocked	uns char		Set to 1, when a section in a locked document is unlocked
cnsPgn	uns char		Chapter number separator for page numbers
fPgnRestart	uns char		Set to 1 when page numbering should be restarted at the beginning of this section
fEndNote	uns char		When 1, footnotes placed at end of section. When 0, footnotes are placed at bottom of page.

Field	Type	Size	Comments
lnc	char		Line numbering code: 0 Per page 1 Restart 2 Continue
grpfIhdtSepOld	char		Specification of which headers and footers are included in this section. See explanation in Headers and Footers topic. No longer used.
nLnnMod	uns short		If 0, no line numbering, otherwise this is the line number modulus (e.g. if nLnnMod is 5, line numbers appear on line 5, 10, etc.)
dxaLnn	long		Distance of
dxaPgn	short		When fAutoPgn ==1, gives the x position of auto page number on page in twips (for Macintosh compatibility only)
dyaPgn	short		When fAutoPgn ==1, gives the y position of auto page number on page in twips (for Macintosh compatibility only)
fLBetween	char		When ==1, draw vertical lines between columns
dmBinFirst	uns short		Bin number supplied from windows printer driver indicating which bin the first page of section will be printed
dmBinOther	uns short		Bin number supplied from windows printer driver indicating which bin the pages other than the first page of section will be printed
dmPaperReq	uns short		dmPaper code for form selected by user
fPropRMark	short		When 1, properties have been changed with revision marking on
ibstPropRMark	short		Index to author IDs stored in hsttbfRMark. used when properties have been changed when revision marking was enabled
dttmPropRMark	DTTM		Date/time at which properties of this were changed for this run of text by the author. (Only recorded when revision marking is on.)
dxtCharSpace	long		How big is a character grid unit (East Asian)
dyaLinePitch	long		Line pitch: How tall a grid unit is up/down
clm	uns short		Grid description: 0=default 1=Chars and line 2=Lines Only 3=Enforce Grid
dmOrientPage	uns char		Orientation of pages in that section. Set to 0 when portrait, 1 when landscape.
iHeadingPgn	uns char		Heading number level for page number
pgnStart	uns short		User specified starting page number
lnnMin	short		Beginning line number for section
pgbProp	short		Page border properties
pgbApplyTo	short	:3	Page border applies to: 0 all pages in this section 1 first page in this section 2 all pages in this section but first 3 whole document (all sections)

Field	Type	Size	Comments
pgbPageDepth	short	:2	Page border depth: 0 in front 1 in back
pgbOffsetFrom	short	:3	Page border offset from: 0 offset from text 1 offset from edge of page
xaPage	uns long		Width of page -- default value is 12240 twips
yaPage	uns long		Height of page -- default value is 15840 twips
xaPageNUp	uns long		Used internally by Word
yaPageNUp	uns long		Used internally by Word
wTextFlow	uns short		Text flow: 0 = Horizontal with no @ fonts 1 = Top to Bottom with @ fonts 2 = Bottom to Top with no @ fonts 3 = Top to Bottom with no @ fonts 4 = Horizontal with @ fonts 5 = Vertical with no @ fonts
dxaLeft	uns long		Left margin -- default value is 1800 twips
dxaRight	uns long		Right margin -- default value is 1800 twips
dyaTop	long		Top margin --default value is 1440 twips
dyaBottom	long		Bottom margin -- default value is 1440 twips
dzaGutter	uns long		Gutter width -- default value is 0 twips
dyaHdrTop	uns long		Y position of top header measured from top edge of page
dyaHdrBottom	uns long		Y position of bottom header measured from top edge of page
ccolM1	short		Number of columns in section - 1
fEvenlySpaced	char		When == 1, columns are evenly spaced. Default value is 1.
vjc	char		Vertical justification code: 0 top justified 1 centered 2 fully justified vertically 3 bottom justified
dxaColumns	long		Distance that will be maintained between columns
fBiDi	uns char		When 1, section direction is right-to-left
fFacingCol	uns char		When 1, section has facing columns
fRTLGutter	uns char		When 1, section has a right-to-left gutter
fRTLAlignment	uns char		When 1, section has right-to-left alignment
rgdxaColumnWidthSpacing array of XA			Array of 89 longs that determine bounds of irregular width columns
dxaColumnWidth	long		Used internally by Word
dmOrientFirst	uns char		Page orientation: 1 Portrait 2 Landscape 3 Mixed
brcTop	BRC		Top page border

Field	Type	Size	Comments
brcLeft	BRC		Left page border
brcBottom	BRC		Bottom page border
brcRight	BRC		Right page border
olstAnm	OLST		Multilevel auto numbering list data (see OLST definition)
fHasOldProps	uchar	1	Used for section property revision marking. The sep at the time fHasOldProps is set to 1, the is the old sep.
rsid	RSDI (long)	4	a random number associated with section formatting which improves the accuracy of Word's document merging.
fpc	FPC	1	Footnote position code 0 print as endnotes 1 print at bottom of page 2 print immediately beneath text
fncFtn	RNC	1	Restart footnote number code 0 don't restart endnote numbering 1 restart for each section 2 restart for each page
epc	epc	1	Endnote position code 0 display endnotes at end of section 3 display endnotes at end of document
rncEdn	rnc	1	Restart endnote number code 0 don't restart endnote numbering 1 restart for each section 2 restart for each page
nFtn	ushort	2	starting footnote number
nfcFtnRef	nfc (short)	2	number format for footnote references See Number Format Table under Document Properties (DOP)
nEdn	ushort	2	starting endnote number
nfcEdnRef	nfc (short)	2	number format for endnote references See Number Format Table under Document Properties (DOP)

The standard SEP is all zeros except as follows:

bkc	2 (new page)
dyaPgn	720 twips (equivalent to .5 in)
dxaPgn	720 twips
fEndnote	1 (True)
fEvenlySpaced	1 (True)
xaPage	12240 twips
yaPage	15840 twips
xaPageNUp	12240 twips
yaPageNUp	15840 twips
dyaHdrTop	720 twips
dyaHdrBottom	720 twips

dmOrientPage	1 (portrait orientation)
dxaColumns	720 twips
dyaTop	1440 twips
dxaLeft	1800 twips
dyaBottom	1440 twips
dxaRight	1800 twips
pgnStart	1

Section Property Exceptions (SEPX)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comment
0	0	cb	uns short			Count of bytes in remainder of SEPX
	2	2	grpprl	char[]		List of sprms that encodes the differences between the properties of a section and Word's default section properties

Shading Descriptor (SHD)

The SHD is a substructure of the CHP, PAP, and TC for Word 2000.

b_{10}	b_{16}	Field	Type	Size	Comments
0	0	cvFore	COLORREF	4	24-bit foreground color
4	4	cvBack	COLORREF	4	24-bit background color
8	8	ipat	uns short	2	Shading pattern (see ipat table below)

Shading Descriptor for Word 97 (SHD80)

The SHD80 is a substructure of the CHP and PAP, and TC for Word 97.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	icoFore	short	:5	001F	Foreground color (see chp.ico)
		icoBack	short	:5	03E0	Background color (see chp.ico)
		ipat	short	:6	FC00	Shading pattern (see ipat table below)

ipat	Pattern
0	Automatic
1	Solid
2	5 Percent
3	10 Percent
4	20 Percent
5	25 Percent
6	30 Percent
7	40 Percent
8	50 Percent
9	60 Percent
10	70 Percent
11	75 Percent
12	80 Percent
13	90 Percent
14	Dark Horizontal
15	Dark Vertical
16	Dark Forward Diagonal
17	Dark Backward Diagonal
18	Dark Cross
19	Dark Diagonal Cross
20	Horizontal
21	Vertical
22	Forward Diagonal
23	Backward Diagonal
24	Cross
25	Diagonal Cross
35	2.5 Percent
36	7.5 Percent
37	12.5 Percent
38	15 Percent
39	17.5 Percent
40	22.5 Percent
41	27.5 Percent
42	32.5 Percent
43	35 Percent

ipat	Pattern
44	37.5 Percent
45	42.5 Percent
46	45 Percent
47	47.5 Percent
48	52.5 Percent
49	55 Percent
50	57.5 Percent
51	62.5 Percent
52	65 Percent
53	67.5 Percent
54	72.5 Percent
55	77.5 Percent
56	82.5 Percent
57	85 Percent
58	87.5 Percent
59	92.5 Percent
60	95 Percent
61	97.5 Percent
62	97 Percent

`cbSHD` (count of bytes of `SHD`) is 2.

Spelling State Mark (SPLS)

The `SPLS` structure is stored in both the `plcfspl` and the `plcfgram`.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	`splf`	short	:4	000F	Spelling result flag: 2 maybe dirty: check if have time 3 dirty: needs to be checked 4 recently edited 5 cannot be checked 7 clean: checked and error-free 10 error 11 repeated word error 12 unknown word error
		`fError`	short	:1	0010	Set to 1 when `SPLS` marks an error
		`fExtend`	short	:1		
		`fTypo`	short	:1		Typo
		`unused`	short	:1		Reserved
		`nCheckLevel`	short	:8	FFE0	ignore-all list version

`cbSPLS` (count of bytes of `SPLS`) is 2

Tab Descriptor (TBD)

The `TBD` is a substructure of the `PAP`.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	jc	byte	:3	07	Justification code 0 left tab 1 centered tab 2 right tab 3 decimal tab 4 bar
		tlc	byte	:3	38	Tab leader code 0 no leader 1 dotted leader 2 hyphenated leader 3 single line leader 4 heavy line leader 5 middle dot
		*	byte	:2	C0	Reserved

cbTBD (count of bytes of TBD) is 1.

Table Cell Descriptors (TC)

The TC is a substructure of the TAP. This structure is never written out to disk but can be built from the appropriate property modifiers. For this reason no offsets into the structure are given.

Field	Type	Size	Bitfield	Comments
fFirstMerged	BF	:1	0001	When 1, cell is first cell of a range of cells that have been merged. When a cell is merged, the display areas of the merged cells are consolidated and the text within the cells is interpreted as belonging to one text stream for purposes of calculating line breaks.
fMerged	BF	:1	0002	When 1, cell has been merged with preceding cell
fVertical	BF	:1	0004	When 1, cell has vertical text flow
fBackward	BF	:1	0008	For a vertical table cell, text flow is bottom to top when 1 and is bottom to top when 0
fRotateFont	BF	:1	0010	When 1, cell has rotated characters (i.e. uses @font)
fVertMerge	BF	:1	0020	When 1, cell is vertically merged with the cell(s) above and/or below. When cells are vertically merged, the display area of the merged cells are consolidated. The consolidated area is used to display the contents of the first vertically merged cell (the cell with fVertRestart set to 1), and all other vertically merged cells (those with fVertRestart set to 0) must be empty. Cells can only be merged vertically if their left and right boundaries are (nearly) identical (i.e. if corresponding entries in rgdxaCenter of the table rows differ by at most 3).
fVertRestart	BF	:1	0040	When 1, the cell is the first of a set of vertically merged cells. The contents of a cell with fVertStart set to 1 are displayed in the consolidated area belonging to the entire set of vertically merged cells. Vertically merged cells with fVertRestart set to 0 must be empty.
vertAlign	BF	:2	0180	Specifies the alignment of the cell contents relative to text flow (e.g. in a cell with bottom to top text flow and bottom vertical alignment, the text is shifted horizontally to match the cell's right boundary): 0 top 1 center 2 bottom

Field	Type	Size	Bitfield	Comments
ftsWidth	BF	:3	0E00	Units for wWidth: 0 Null 1 Auto, ignores wWidth 2 Percentage (in 50ths of a percent) 3 Twips
fFitText	BF	:1	1000	When 1, make the text fit the table cell
fNoWrap	BF	:1	2000	When 1, do not allow text to wrap in the table cell
fUnused	BF	:2	C000	Not used
wWidth	short			Preferred cell width
shd	SHD	10		Cell shading
tcd	TCD	16		Diagonal Cell border information
mpibrcwCellPadding[cbrcTc]	short	8		Array of cell margins, convenient indexing of the following four properties
wCellPaddingLeft	short	2		Left cell margin/padding
wCellPaddingTop	short	2		Top cell margin/padding
wCellPaddingBottom	short	2		Bottom cell margin/padding
wCellPaddingRight	short	2		Right cell margin/padding
mpibrcftsCellPadding[cbrcTc]	uchar	4		Array of cell margin units, convenient indexing of the following four properties
ftsCellPaddingLeft	uchar	1		Left cell margin/padding units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellPaddingTop	uchar	1		Top cell margin/padding units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellPaddingBottom	uchar	1		Bottom cell margin/padding units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellPaddingRight	uchar	1		Right cell margin/padding units: 0 Null 1 – 2 Not relevant 3 Twips
mpibrcwCellSpacing[cbrcTc]	short	8		Array of cell spacing, convenient indexing of the following four properties
wCellSpacingLeft	short	2		Left cell spacing
wCellSpacingTop	short	2		Top cell spacing
wCellSpacingBottom	short	2		Bottom cell spacing
wCellSpacingRight	short	2		Right cell spacing
mpibrcftsCellSpacing[cbrcTc]	uchar	4		Array of cell spacing units, convenient indexing of the following four properties
ftsCellSpacingLeft	uchar	1		Left cell spacing units: 0 Null 1 – 2 Not relevant 3 Twips

Field	Type	Size	Bitfield	Comments
ftsCellSpacingTop	uchar	1		Top cell spacing units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellSpacingBottom	uchar	1		Bottom cell spacing units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellSpacingRight	uchar	1		Right cell spacing units: 0 Null 1 – 2 Not relevant 3 Twips
mpibrcfValidGapHalf[cbrcTc]	uchar	4		Used internally by Word
mpibrcdzaGapHalf[cbrcTc]	short	8		Used internally by Word
rgbrc[cbrcTc]	BRC	32		Array of borders; cbrcTc = 4
brcTop	BRC	8		Top border
brcLeft	BRC	8		Left border
brcBottom	BRC	8		Bottom border
brcRight	BRC	8		Right border

Table Cell Diagonal Borders (TCD)

The TCD is a substructure of the TC. This structure is never written out to disk but can be built from the appropriate property modifiers. For this reason no offsets into the structure are given.

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	brcTL2BR	BRC			Diagonal border from the top left to the bottom right of the cell.
8	8	brcTR2BL	BRC			Diagonal border from the top right to the bottom left of the cell.

Table Autoformat Look sPecifier (TLP)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	itl	short			Index to Word's table of table looks (see itl table below)
2	2	fBorders	short	:1	0001	When ==1, use the border properties from the selected table look
		fShading	short	:1	0002	When ==1, use the shading properties from the selected table look
		fFont	short	:1	0004	When ==1, use the font from the selected table look
		fColor	short	:1	0008	When ==1, use the color from the selected table look
		fBestFit	short	:1	0010	When ==1, do best fit from the selected table look
		fHdrRows	short	:1	0020	When ==1, apply properties from the selected table look to the header rows in the table
		fLastRow	short	:1	0040	When ==1, apply properties from the selected table look to the last row in the table

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
		fHdrCols	short	:1	0080	When ==1, apply properties from the selected table look to the header columns of the table
		fLastCol	short	:1	0100	When ==1, apply properties from the selected table look to the last column of the table

itl	Table look
0	(none)
1	Simple 1
2	Simple 2
3	Simple 3
4	Classic 1
5	Classic 2
6	Classic 3
7	Classic 4
8	Colorful 1
9	Colorful 2
10	Colorful 3
11	Columns 1
12	Columns 2
13	Columns 3
14	Columns 4
15	Columns 5
16	Grid 1
17	Grid 2
18	Grid 3
19	Grid 4
20	Grid 5
21	Grid 6
22	Grid 7
23	Grid 8
24	List 1
25	List 2
26	List 3
27	List 4
28	List 5
29	List 6
30	List 7
31	List 8
32	3D Effects 1
33	3D Effects 2
34	3D Effects 3
35	Contemporary
36	Elegant
37	Professional

itl	Table look
38	Subtle1
39	Subtle2

`cbTLP` (count of bytes of `TLP`) is 4.

Table Properties (TAP)

This structure is never written out to disk but can be built from the appropriate property modifiers. For this reason no offsets into the structure are given.

Field	Type	Size	Comments
istd	short	2	Table style for the Table
jc	short	2	Justification code. specifies how table row should be justified within its column. 0 left justify 1 center 2 right justify
dxaGapHalf	long	4	Measures half of the white space that will be maintained between text in adjacent columns of a table row. A dxaGapHalf width of white space will be maintained on both sides of a column boundary.
dyaRowHeight	long	4	When greater than 0. guarantees that the height of the table will be at least dyaRowHeight high. When less than 0, guarantees that the height of the table will be exactly absolute value of dyaRowHeight high. When 0, table will be given a height large enough to represent all of the text in all of the cells of the table. Cells with vertical text flow make no contribution to the computation of the height of rows with auto or at least height. Neither do vertically merged cells, except in the last row of the vertical merge. If an auto height row consists entirely of cells which have vertical text direction or are vertically merged, and the row does not contain the last cell in any vertical cell merge, then the row is given height equal to that of the end of cell mark in the first cell.
fCantSplit	uchar	1	When 1, table row may not be split across page bounds
fCantSplit90	uchar	1	When 1, table row may not be split across page bounds. Used for Word 2000 and Word 97.
fTableHeader	uchar	1	When 1, table row is to be used as the header of the table
tlp	TLP	4	Table look specifier (see TLP definition)
wWidth	short	2	Preferred table width
wWidthIndent	short	2	Left Indent
wWidthBefore	short	2	Width of invisible cell (used for layout purposes) before the first visible cell in the row.
wWidthAfter	short	2	Width of invisible cell (used for layout purposes) after the last visible cell in the row.
fAutofit	unsigned short	:1	When set to 1, AutoFit this table

Field	Type	Size	Comments
istd	short	2	Table style for the Table
fKeepFollow	unsigned short	:1	When set to 1, keep this row with the following row
ftsWidth	unsigned short	:3	Units for wWidth: 0 Null 1 Auto, ignores wWidth 2 Percentage (in 50ths of a percent) 3 Twips
ftsWidthIndent	unsigned short	:3	Units for wWidthIndent: 0 Null 1 Auto, ignores wWidthIndent 2 Percentage (in 50ths of a percent) 3 Twips
ftsWidthBefore	unsigned short	:3	Units for wWidthBefore: 0 Null 1 Auto, ignores wWidthBefore 2 Percentage (in 50ths of a percent) 3 Twips
ftsWidthAfter	unsigned short	:3	Units for wWidthAfter: 0 Null 1 Auto, ignores wWidthAfter 2 Percentage (in 50ths of a percent) 3 Twips
fNeverBeenAutofit	unsigned short	:1	When 1, table has never been autofit
fInvalAutofit	unsigned short	:1	When 1, TAP is still valid, but autofit properties aren't
empty	unsigned short	:3	Not used
fVert	unsigned short	:1	When 1, positioned in vertical text flow
pcVert	unsigned short	:2	Vertical position code. Specifies coordinate frame to use when paragraphs are absolutely positioned. 0 vertical position coordinates are relative to margin 1 coordinates are relative to page 2 coordinates are relative to text. This means: relative to where the next non-APO text would have been placed if this APO did not exist.
pcHorz	unsigned short	:2	Horizontal position code. Specifies coordinate frame to use when paragraphs are absolutely positioned. 0 horizontal position coordinates are relative to column. 1 coordinates are relative to margin 2 coordinates are relative to page
empty	unsigned short	:8	Not used
dxaAbs	long	4	Absolute horizontal position
dyaAbs	long	4	Absolute vertical position
dxaFromText	long	4	Left distance from surrounding text when absolutely positioned
dyaFromText	long	4	Top distance from surrounding text when absolutely positioned

Field	Type	Size	Comments
istd	short	2	Table style for the Table
dxaFromTextRight	long	4	Right distance from surrounding text when absolutely positioned
dyaFromTextBottom	long	4	Bottom distance from surrounding text when absolutely positioned
fBiDi	uchar	1	When 1, table is right-to-left. Logical right-to-left table: The CP stream of a right-to-left table is meant to be displayed from right to left. So for example the first table cell is displayed on the right side of the table instead of the left.
fRTL	uchar	1	Word 2000 style right-to-left table. Visual right-to-left table: The CP stream of a right-to-left table is displayed from left to right just as for a "normal" table. So, the text which is meant to be in the "first" (rightmost) table cell must be placed in the last table cell in the CP stream.
fNoAllowOverlap	uchar	1	When set to 1, do not allow absolutely positioned table to overlap with other tables
fSpare	uchar	1	Not used
grpfTap	unsigned short	2	Used internally by Word
fFirstRow	unsigned short	:1	Used internally by Word: first row
fLastRow	unsigned short	:1	Used internally by Word: last row
fOutline	unsigned short	:1	Used internally by Word: row was cached for outline mode
fOrigWordTableRules	unsigned short	:1	Used internally by Word: table combining like Word 5.x for the Macintosh and WinWord 1.x
fCellSpacing	unsigned short	:1	Used internally by Word: When set to 1 cell spacing is allowed
unused	unsigned short	:11	Not used
itcMac	short	2	Count of cells defined for this row. itcMac must be >= 0 and less than or equal to 64.
dxaAdjust	long	4	Used internally by Word
dxaWebView	long	4	Used internally by Word
dxaRTEWrapWidth	long	4	Used internally by Word
dxaColWidthWwd	long	4	Used internally by Word
pctWwd	short	2	Used internally by Word: percent of Window size for AutoFit in WebView
fWrapToWwd	unsigned short	:1	Used internally by Word: Wrap to window is on when set to 1

Field	Type	Size	Comments
istd	short	2	Table style for the Table
fNotPageView	unsigned short	:1	Used internally by Word: when set to 1 we are not in Page View
unused	unsigned short	:1	Not used
fWebView	unsigned short	:1	Used internally by Word: Web View is on when set to 1
fAdjusted	unsigned short	:1	Used internally by Word
unused	unsigned short	:11	Not used
rgdxaCenter[itcMax+1]	Short[itcMax+1]	130	rgdxaCenter[0] is the left boundary of cell 0 measured relative to margin rgdxaCenter[tap.itcMac – 1] is left boundary of last cell rgdxaCenter[tap.itcMac] is right boundary of last cell.
rgdxaCenterPrint[itcMax+1]	short[itcMax+1]	130	Used internally by Word
shdTable	SHD	10	Table shading
rgbrcTable[ibrcTableMax]	BRC[ibrcTableMax]	48	Array of borders; ibrcTableMax==6
mpibrcwCellPaddingDefault[cbrcTc]	short	8	Array of default cell margins. Index into the next four properties.
wCellPaddingDefaultTop	short	2	Default left cell margin/padding
wCellPaddingDefaultLeft	short	2	Default top cell margin/padding
wCellPaddingDefaultBottom	short	2	Default Bottom cell margin/padding
wCellPaddingDefaultRight	short	2	Default right cell margin/padding
mpibrcftsCellPaddingDefault[cbrcTc]	uchar	4	Array of default cell margin units. Index into the next four properties
ftsCellPaddingDefaultTop	uchar	1	Default left cell margin/padding units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellPaddingDefaultLeft	uchar	1	Default top cell margin/padding units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellPaddingDefaultBottom	uchar	1	Default bottom cell margin/padding units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellPaddingDefaultRight	uchar	1	Default right cell margin/padding units: 0 Null 1 – 2 Not relevant 3 Twips
mpibrcwCellSpacingDefault[cbrcTc]	short	8	Array of default cell spacing. Index into the next four properties
wCellSpacingDefaultTop	short	2	Default left cell spacing
wCellSpacingDefaultLeft	short	2	Default top cell spacing
wCellSpacingDefaultBottom	short	2	Default Bottom cell spacing

Field	Type	Size	Comments
istd	short	2	Table style for the Table
wCellSpacingDefaultRight	short	2	Default right cell spacing
mpibrcftsCellSpacingDefault[cbrcTc]	uchar	4	Array of default cell spacing units. Index into the next four properties
ftsCellSpacingDefaultTop	uchar	1	Default left cell spacing units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellSpacingDefaultLeft	uchar	1	Default top cell spacing units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellSpacingDefaultBottom	uchar	1	Default bottom cell spacing units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellSpacingDefaultRight	uchar	1	Default right cell spacing units: 0 Null 1 – 2 Not relevant 3 Twips
mpibrcwCellPaddingOuter[cbrcTc]	short	8	Array of default outer cell margins. Index into the next four properties.
wCellPaddingOuterTop	short	2	Default outer left cell margin/padding
wCellPaddingOuterLeft	short	2	Default outer top cell margin/padding
wCellPaddingOuterBottom	short	2	Default outer bottom cell margin/padding
wCellPaddingOuterRight	short	2	Default outer right cell margin/padding
mpibrcftsCellPaddingOuter[cbrcTc]	uchar	4	Array of default outer cell margin units
ftsCellPaddingOuterTop	uchar	1	Default outer left cell margin/padding units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellPaddingOuterLeft	uchar	1	Default outer top cell margin/padding units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellPaddingOuterBottom	uchar	1	Default outer bottom cell margin/padding units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellPaddingOuterRight	uchar	1	Default outer right cell margin/padding units: 0 Null 1 – 2 Not relevant 3 Twips
mpibrcwCellSpacingOuter[cbrcTc]	short	8	Array of default outer cell spacing. Index into the next four properties.
wCellSpacingOuterTop	short	2	Default outer left cell spacing
wCellSpacingOuterLeft	short	2	Default outer top cell spacing
wCellSpacingOuterBottom	short	2	Default outer Bottom cell spacing

Field	Type	Size	Comments
istd	short	2	Table style for the Table
wCellSpacingOuterRight	short	2	Default outer right cell spacing
mpibrcftsCellSpacingOuter[cbrcTc]	uchar	4	Array of default outer cell spacing units. Index into the next four properties.
ftsCellSpacingOuterTop	uchar	1	Default outer left cell spacing units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellSpacingOuterLeft	uchar	1	Default outer top cell spacing units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellSpacingOuterBottom	uchar	1	Default outer bottom cell spacing units: 0 Null 1 – 2 Not relevant 3 Twips
ftsCellSpacingOuterRight	uchar	1	Default outer right cell spacing units: 0 Null 1 – 2 Not relevant 3 Twips
rgtc[itcMax]	TC[itcMax]	6144	Array of table cells descriptors (TCs); itcMax = 64
rgshd	SHD80[itcMax]		Array of cell shades. Only applies to Word 97. In Word 2000 and later versions, table shades are stored inside the TC structures in rgtc[].
fPropRMark	uchar	1	Set to 1 if property revision
ibstPropRMark	IBST	2	Index to author table for property change
dttmPropRMark	DTTM	4	Date/Time of property change
fHasOldProps	uchar	1	Has old properties
ipgp	uns long	4	HTML DIV ID for this table
hplcnf	HPL	4	Pointer to conditional table properties
rsid	RSID	4	Save ID for last time this TAP was revised
tcDefault	TC		This TC is only used in styles; at style apply time, its values will get propagated to the appropriate rgtc locations
cHorzBands	uns char	1	Size of each horizontal style band, in number of rows
cVertBands	uns char	1	Size of a vertical style band, in number of columns
shdTableDef	SHD	10	Default shading for the table
rgbrcInsideDefault[0]	BRC	8	Border definition for inside horizontal borders
rgbrcInsideDefault[1]	BRC	8	Border definition for inside vertical borders

TeXtBoX Story (FTXBXS)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	cTxbx	long			When not fReusable, counts the number of textboxes in this story chain
0	0	iNextReuse	long			When fReusable, the index of the next in the linked list of reusable FTXBXSs
4	4	cReusable	long			If fReusable, counts the number of reusable FTXBXSs follow this one in the linked list
8	8	fReusable	short			This FTXBXS is not currently in use
10	A		long			Reserved
14	E	lid	long			Shape identifier (see FSPA) for first Office shape in textbox chain.
18	12	txidUndo	long			

cbFTXBXS (count of bytes of FTXBXS) is 22 (decimal), 16 (hex).

WorK Book (WKB)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	fn	short			
2	2	grfwkb	uns short			
4	4	lvl	short			
6	6	fnpt	short	:4	000F	
		fnpd	short	:12	FFF0	
8	8	doc	long			Unused

cbWKB (count of bytes of WKB) is 12 (decimal), C (hex).

Information Rights Management (IRM)

Rights Management protected documents are encrypted in the same way that OfficeXP documents are encrypted using a password.

This means:

- The macro streams in the document are not encrypted
- The Document Summary Information stream is not encrypted

Rights Management protected documents are in essence two documents in one. One is a traditional backwards-compatible document consisting of fixed text informing the reader that they need a later version of Office to access the protected content. This is stored as text and an image in the "Data" stream, where unprotected document content would otherwise be stored. The other, not previously found in Office documents, is an encrypted copy of the document content, along with new information required to support IRM protection. The details of the differences are discussed in the sections that follow. An overview of Information rights Management can be found at http://download.microsoft.com/download/a/4/2/a4262821-6f21-450f-85d3-ebbba001a6ef/How%20to%20Use%20Information%20Rights%20Management.doc.

DataSpaces

Every rights managed file contains a new storage named "\006DataSpaces" which contains meta information used to help manage the process of protecting the content within the document.More information can be found at http://msdn2.microsoft.com/en-us/library/aa767782(VS.85).aspx. The most important content in this storage is the information under the "TransformInfo" storage. This storage contains the issuance licenses and end-user licenses required to protect and open a rights managed file.

DRMContent

The new stream named "\011DRMContent" contains the encrypted binary content of the Word document. The format of this stream contains a series of encrypted bytes. When you decrypt the whole stream and open the resultant byte stream as a compound storage, then that storage will contain all streams and substorages that are found in a normal Word document, using the exact same binary file format as a non-IRM-protected Word file. Only the encrypted streams are located inside of this storage. The unencrypted streams (for example Document Summary Information and the macro stream in Word and Microsoft Office Excel®) are not stored inside this storage. They are found unencrypted off of the root of the document's storage.

DRMViewerContent

The final new stream that may exist within an IRM-protected file is the optional "\011DRMViewerContent" stream which contains a compressed, encrypted MHTML stream for users of the Rights Management Add-on for Internet Explorer. This is the option for users who need to see IRM protected content but do not have access to an IRM enabled Office client.

Appendix A - Reading a Macintosh PICT Graphic

As described under "picture" in the Definition section of this document, some pictures in Word documents are stored as Macintosh PICT graphics, particularly in files created by Word for the Macintosh. All pictures, including these, are stored as a block of binary data attached to a special `chPic` character in the text stream. This block always begins with a `PIC` structure. (Please see the "picture" definition mentioned above for more information on general picture-reading.)

Normal graphics follow the `PIC` structure with a single Office shape, Windows metafile, bitmap, or TIFF representation, as described in the "picture" definition section. Macintosh PICT graphics have a standard, unchanging Windows metafile after the `PIC` which always depicts an "x", followed by the actual Macintosh PICT picture. This is for backward-compatibility with older readers, which expect to find a Windows metafile after the `PIC` structure. These readers will simply display the fixed "x" image. In the Macintosh PICT case, the `PIC` structure's `lcb` field represents the size of the entire picture data block, including the `PIC` itself, the "x" metafile and the Macintosh PICT data. (see the description of the `PIC` structure in the Structure Definitions section of this document.)

To distinguish between normal and Macintosh PICT graphics, a reader needs to detect the presence of the special "x" metafile. The bytes below are in an early portion of the "x" metafile.

```
unsigned char rgbWmfXBegin[] =
{
   '\x14', '\x00', '\x00', '\x00', '\x26', '\x06', '\x0F', '\x00', '\x1E', '\x00',
   '\xFF', '\xFF', '\xFF', '\xFF', '\x04', '\x00', '\x14', '\x00', '\x00', '\x00',
   '\x57', '\x6F', '\x72', '\x64', '\x0E', '\x00', '\x4D', '\x69', '\x63', '\x72',
   '\x6F', '\x73', '\x6F', '\x66', '\x74', '\x20', '\x57', '\x6F', '\x72', '\x64',
   '\x0E', '\x00', '\x00', '\x00', '\x26', '\x06', '\x0F', '\x00', '\x12', '\x00',
   '\x57', '\x6F', '\x72', '\x64', '\xFF', '\xFF', '\x08', '\x00', '\x00', '\x00'

   /* "x" wmf and PICT data sizes immediately follow as 2 four-byte longs */
};

#define cbMETAHDR 18 // size of a standard Windows metafile header
#define cbWmfXBegin 60  // length of this beginning section of the x metafile
```

After reading the `PIC` structure from the picture data block, the reader should skip `cbMETAHDR` bytes (the size of a standard Windows metafile header). It should then compare the next `cbWmfXBegin` bytes in the picture data block against the bytes in the `rgbWmfXBegin` array above. If they do not match, the picture is a normal picture—Windows metafile, bitmap or TIFF.

If they *do* match, then the reader should read the next 8 bytes in the picture data block as two 4-byte "long"s (Intel 80x86 byte order). These numbers are the sizes (in bytes) of the "x" metafile and the Macintosh PICT data, respectively. The size of the "x" metafile is measured from its start immediately after the `PIC` structure. It is possible for the PICT's size to be zero. In this case, there is no PICT data, and the reader may use the "x" Windows metafile as the picture's representation.

Appendix B – Calculation of font (FTC) and language (LID)

Certain Unicode characters are shared between East Asian and non-East Asian scripts requiring the calculation of font and language, based on the Unicode character code and the `chp.idctHint` property.

Characters are classified into one of four groups, ASCII, East Asian, floating, and non-East Asian. Properties are calculated as follows:

Character type	Font (ftc)	Language (lid)
ASCII	`sprmCRgftc0`	`sprmCRglid0`
non-East Asian	`sprmCRgftc2`	`sprmCRglid0`
East Asian	`sprmCRgftc1`	`sprmCRglid1`
shared character	`sprmCRgftc2` if `chp.idctHint==0` `sprmCRgftc1` if `chp.idctHint==1`	`sprmCRglid0` if `chp.idctHint==0` `sprmCRglid1` if `chp.idctHint==1`

The table below defines the classification of various ranges of Unicode characters:

Unicode subrange	Character range	Classification
usrBasicLatin	0x20->0x7f	ASCII
usrLatin1	0xa0->0xff	Some shared (see notes below)
usrLatinXA	0x100->0x17f	Some shared (see notes below)
usrLatinXB	0x180->0x24f	Some shared (see notes below)
usrIPAExtensions	0x250->0x2af	Some shared (see notes below)
usrSpacingModLetters	0x2b0->0x2ff	Shared
usrCombDiacritical	0x300->0x36f	Shared
usrBasicGreek	0x370->0x3cf	Shared
usrGreekSymbolsCop	0x3d0->0x3ff	Non-East Asian
usrCyrillic	0x400->0x4ff	Shared
usrArmenian	0x500->0x58f	Non-East Asian
usrBasicHebrew	0x5d0->0x5ff	Non-East Asian
usrHebrewXA	0x590->0x5cf	Non-East Asian
usrBasicArabic	0x600->0x652	Non-East Asian
usrArabicX	0x653->0x6ff	Non-East Asian
usrDevangari	0x900->0x97f	Non-East Asian
usrBengali	0x980->0x9ff	Non-East Asian
usrGurmukhi	0xa00->0xa7f	Non-East Asian
usrGujarati	0xa80->0xaff	Non-East Asian
usrOriya	0xb00->0xb7f	Non-East Asian
usrTamil	0x0b80->0x0bff	Non-East Asian

Unicode subrange	Character range	Classification
usrTelugu	0x0c00->0x0c7f	Non-East Asian
usrKannada	0x0c80->0x0cff	Non-East Asian
usrMalayalam	0x0d00->0x0d7f	Non-East Asian
usrThai	0x0e00->0x0e7f	Non-East Asian
usrLao	0x0e80->0x0eff	Non-East Asian
usrBasicGeorgian	0x10d0->0x10ff	Non-East Asian
usrGeorgianExtended	0x10a0->0x10cf	Non-East Asian
usrHangulJamo	0x1100->0x11ff	Non-East Asian
usrLatinExtendedAdd	0x1e00->0x1eff	Shared
usrGreekExtended	0x1f00->0x1fff	Non-East Asian
usrGeneralPunct	0x2000->0x206f	Shared
usrSuperAndSubscript	0x2070->0x209f	Shared
usrCurrencySymbols	0x20a0->0x20cf	Shared
usrCombDiacriticsS	0x20d0->0x20ff	Shared
usrLetterlikeSymbols	0x2100->0x214f	Shared
usrNumberForms	0x2150->0x218f	Shared
usrArrows	0x2190->0x21ff	Shared
usrMathematicalOps	0x2200->0x22ff	Shared
usrMiscTechnical	0x2300->0x23ff	Shared
usrControlPictures	0x2400->0x243f	Shared
usrOpticalCharRecog	0x2440->0x245f	Shared
usrEnclosedAlphanum	0x2460->0x24ff	Shared
usrBoxDrawing	0x2500->0x257f	Shared
usrBlockElements	0x2580->0x259f	Shared
usrGeometricShapes	0x25a0->0x25ff	Shared
usrMiscDingbats	0x2600->0x26ff	Shared
usrDingbats	0x2700->0x27bf	Shared
usrCJKSymAndPunct	0x3000->0x303f	East Asian
usrHiragana	0x3040->0x309f	East Asian
usrKatakana	0x30a0->0x30ff	East Asian
usrBopomofo	0x3100->0x312f	East Asian
usrHangulCompatJamo	0x3130->0x318f	East Asian
usrCJKMisc	0x3190->0x319f	East Asian
usrEnclosedCJKLtMnth	0x3200->0x32ff	East Asian

Unicode subrange	Character range	Classification
usrCJKCompatibility	0x3300->0x33ff	East Asian
	0x4a00->0x4dff	
usrCJKUnifiedIdeo	0x4e00->0x9fff	East Asian
usrHangul	0xac00->0xd7a3	East Asian
usrReserved1	0xd800	
usrReserved2		
usrPrivateUseArea	0xe000->0xf8ff	Shared
usrCJKCompatibilityIdeographs	0xf900->0xfaff	East Asian
usrAlphaPresentationForms	0xfb00->0xfb4f	Shared
usrArabicPresentationFormsA	0xfb50->0xfdff	Shared
usrCombiningHalfMarks	0xfe20->0xfe2f	East Asian
usrCJKCompatForms	0xfe30->0xfe4f	East Asian
usrSmallFormVariants	0xfe50->0xfe6f	East Asian
usrArabicPresentationFormsB	0xfe70->0xfefe	Shared
usrHFWidthForms	0xff00->0xffef	East Asian
usrSpecials	0xfff0->0xfffd	Non-East Asian

The table below describes the behavior of the Unicode subrange `usrLatin1`. Shared characters are marked in this table with a 1, while characters marked with a 0 are considered "non-East Asian". All other characters in this Unicode subrange are considered "non-East Asian".

```
// 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
   0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, // 0x00a0-0x00af
   1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, // 0x00b0-0x00bf
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, // 0x00c0-0x00cf
   0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, // 0x00d0-0x00df
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, // 0x00e0-0x00ef
   0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, // 0x00f0-0x00ff
   };
```

The table below describes the behavior of the Unicode range `usrLatinXA`. Shared characters are marked in this table with a 1, while characters marked with a 0 are considered "non-East Asian". All other characters in this Unicode subrange are considered "non-East Asian".

```
// 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
   1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, // 0x0100-0x010f
   0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, // 0x0110-0x011f
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, // 0x0120-0x012f
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, // 0x0130-0x013f
   0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, // 0x0140-0x014f
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, // 0x0150-0x015f
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, // 0x0160-0x016f
```

In `usrLatinXB` shared characters are 0x192, 0x1FA, 0x1FB, 0x1FC, 0x1FD, 0x1FE and 0x1FF. All other characters in this Unicode subrange are considered "non-East Asian".

In `usrIPAExtensions` shared characters are 0x251, and 0x261.

An optimization is available. If the East Asian font `chp.ftcFE=0` and `chp.idctHint=0` and `chp.ftcAscii=chp.ftcOther`, the font is `chp.ftcAscii` and the language is `chp.lidDefault`.