**In plain-ish English**

1. Get WordDocument stream
2. Read FIB from offset 0 (**154 bytes** excluding FibRgLcb97)
3. Get FibRgFcLcb97 (**185\*4** bytes)
   1. Get fsClx to find location of Clx (**66\*4** bytes)
   2. Get lcbCLx to find size of Clx (**67\*4** bytes)
4. Read Clx from Table Stream
   1. Iterate through RgPrc variable to find 0-n Prcs. Once RgPrc = 0x02, we’ve found Pcdt
5. Read Pcdt from Clx using 4a offset
   1. Read lcb to get size of PlcPcd
6. Read PlcPcd
7. Calculate the largest i such that PlcPcd.aCp[i] ≤ cp. The aCp array has one more element than the aPcd array
8. Grab all the characters from the aCp array

**Selected portions of the spec**

**2.4.1 Retrieving Text**

The following algorithm specifies how to find the text at a particular character position (cp). Negative character positions are not valid.

1. Read the **FIB** from offset zero in the **WordDocument** Stream.
2. All versions of the FIB contain exactly one **FibRgFcLcb97**, though it can be nested in a larger structure. FibRgFcLcb97.**fcClx** specifies the offset in the Table Stream of a Clx. FibRgFcLcb97.lcbClx specifies the size, in bytes, of that Clx. Read the **Clx** from the **Table** Stream.
3. The Clx contains a **Pcdt**, and the Pcdt contains a **PlcPcd**. Find the largest i such that PlcPcd.aCp[i] ≤ cp. As with all Plcs, the elements of PlcPcd.aCp are sorted in ascending order. Recall from the definition of a Plc that the aCp array has one more element than the aPcd array. Thus, if the last element of PlcPcd.aCp is less than or equal to cp, cp is outside the range of valid character positions in this document.
4. PlcPcd.aPcd[i] is a **Pcd**. Pcd.fc is an **FcCompressed** that specifies the location in the WordDocument Stream of the text at character position PlcPcd.aCp[i].
5. If FcCompressed.fCompressed is zero, the character at position cp is a 16-bit Unicode character at offset FcCompressed.fc + 2(cp - PlcPcd.aCp[i]) in the WordDocument Stream. This is to say that the text at character position PlcPcd.aCP[i] begins at offset FcCompressed.fc in the WordDocument Stream and each character occupies two bytes.
6. If FcCompressed.fCompressed is 1, the character at position cp is an 8-bit ANSI character at offset (FcCompressed.fc / 2) + (cp - PlcPcd.aCp[i]) in the WordDocument Stream, unless it is one of the special values in the table defined in the description of FcCompressed.fc. This is to say that the text at character position PlcPcd.aCP[i] begins at offset FcCompressed.fc / 2 in the WordDocument Stream and each character occupies one byte.

**2.5.1 Fib**

The Fib structure contains information about the document and specifies the file pointers to various portions that make up the document.

The Fib is a variable length structure. With the exception of the base portion which is fixed in size, every section is preceded with a count field that specifies the size of the next section.

**base (32 bytes):** The FibBase.

**csw (2 bytes):** An unsigned integer that specifies the count of 16-bit values corresponding to fibRgW that follow. MUST be 0x000E.

**fibRgW (28 bytes):** The FibRgW97.

**cslw (2 bytes):** An unsigned integer that specifies the count of 32-bit values corresponding to fibRgLw that follow. MUST be 0x0016.

**fibRgLw (88 bytes):** The FibRgLw97.

**cbRgFcLcb (2 bytes):** An unsigned integer that specifies the count of 64-bit values corresponding to fibRgFcLcbBlob that follow. This MUST be one of the following values, depending on the value of nFib.

|  |  |
| --- | --- |
| **Value of nFib** | **cbRgFcLcb** |
| 0x00C1 | 0x005D |
| 0x00D9 | 0x006C |
| 0x0101 | 0x0088 |
| 0x010C | 0x00A4 |
| 0x0112 | 0x00B7 |

**fibRgFcLcbBlob (variable):** The FibRgFcLcb.

**2.5.6 FibRgFcLcb97**

The FibRgFcLcb97 structure is a variable-length portion of the Fib.

**(66\*4+154?) fcClx (4 bytes):** An unsigned integer that specifies an offset in the Table Stream. A Clx begins at this offset.

**(67\*4+154?) lcbClx (4 bytes):** An unsigned integer that specifies the size, in bytes, of the Clx at offset fcClx in the Table Stream. This value MUST be greater than zero.

**2.9.38 Clx**

The Clx structure is an array of zero, 1, or more Prcs followed by a Pcdt.

**RgPrc (variable**): An array of Prc. If this array is empty, the first byte of the Clx MUST be 0x02. 0x02 is invalid as the first byte of a Prc, but required for the Pcdt.

**Pcdt (variable):** A Pcdt.

**2.9.209 Prc**

The **Prc** structure specifies a set of properties for document content that is referenced by a **Pcd** structure.

**clxt (1 byte):**  This value MUST be 0x01.

**data (variable):** A PrcData that specifies a set of properties.

**2.9.210 PrcData**

The **PrcData** structure specifies an array of **Prl** elements and the size of the array.

**cbGrpprl (2 bytes):**  A signed integer that specifies the size of **GrpPrl**, in bytes. This value MUST be less than or equal to 0x3FA2.

**GrpPrl (variable):** An array of **Prl** elements. **GrpPrl** contains a whole number of **Prl** elements.

**2.9.178 Pcdt**

The Pcdt structure contains a PlcPcd structure and specifies its size.

**clxt (1 byte):** This value MUST be 0x02.

**lcb (4 bytes):** An unsigned integer that specifies the size, in bytes, of the PlcPcd structure.

**PlcPcd (variable):** A PlcPcd structure. As with all Plc elements, the size that is specified by lcb MUST result in a whole number of Pcd structures in this PlcPcd structure.

**2.8.35 PlcPcd**

The PlcPcd structure is a PLC whose data elements are Pcds (8 bytes each). A PlcPcd MUST NOT contain duplicate CPs.

**aCP (variable):** An array of CPs that specifies the starting points of text ranges. The end of each range is the beginning of the next range. All CPs MUST be greater than or equal to zero. If any of the fields ccpFtn, ccpHdd, ccpMcr, ccpAtn, ccpEdn, ccpTxbx, or ccpHdrTxbx from FibRgLw97 are nonzero, then the last CP MUST be equal to the sum of those fields plus ccpText+1. Otherwise, the last CP MUST be equal to ccpText.

**aPcd (variable):** An array of Pcds (8 bytes each) that specify the location of text in the WordDocument stream and any additional properties of the text. If aPcd[i].fc.fCompressed is 1,then the byte offset of the last character of the text referenced by aPcd[i] is given by the following.



Otherwise, the byte offset of the last character of the text referenced by aPcd[i] is given by the following.



Because aCP MUST be sorted in ascending order and MUST NOT contain duplicate CPs, (aCP[i+1]-aCP[i])>0, for all valid indexes i of aPcd. Because a PLC MUST contain one more CP than a data element, i+1 is a valid index of aCP if i is a valid index of aPcd.

**2.2.1 Character Position** **(CP)**

A character position, which is also known as a CP, is an unsigned 32-bit integer that serves as the zero-based index of a character in the document text. There is no requirement that the text at consecutive character positions be at adjacent locations in the file. The size of each character in the file also varies. The location and size of each character in the file can be computed using the algorithm in section 2.4.1 (Retrieving Text).

Characters include the text of the document, anchors for objects such as footnotes or textboxes, and control characters such as paragraph marks and table cell marks.

Unless otherwise specified by a particular usage, a CP MUST be greater than or equal to zero and less than 0x7FFFFFFF. The range of valid character positions in a particular document is given by the algorithm in section 2.4.1 (Retrieving Text).

**2.2.2 PLC**

The PLC structure is an array of character positions followed by an array of data elements. The data elements for any PLC MUST be the same size of zero or more bytes. The number of CPs MUST be one more than the number of data elements. The CPs MUST appear in ascending order. There are different types of PLC structures, as specified in section 2.8. Each type specifies whether duplicate CPs are allowed for that type.

If the total size of a PLC (cbPlc) and the size of a single data element (cbData) are known, the number of data elements in that PLC (n) is given by the following expression:



The preceding expression MUST yield a whole number for n.

**aCP (variable length):** An array of CP elements. Each type of PLC structure specifies the meaning of the CP elements and the allowed range.

**aData (variable length):** Each type of PLC structure specifies the structure and meaning of the data elements, any restrictions on the number of data elements, and any restrictions on the data contained therein. It also specifies the relationship between the data elements and the corresponding CPs.

**2.9.177 Pcd**

The Pcd structure specifies the location of text in the WordDocument Stream and additional properties for this text. A Pcd structure is an element of a PlcPcd structure.

**A - fNoParaLast (1 bit):** If this bit is 1, the text MUST NOT contain a paragraph mark.

**B - fR1 (1 bit):** This field is undefined and MUST be ignored.

**C - fDirty (1 bit):** This field MUST be 0.

**fR2 (13 bits):** This field is undefined and MUST be ignored.

**fc (4 bytes):** An **FcCompressed** structure that specifies the location of the text in the WordDocument Stream.

**prm (2 bytes):** A Prm structure that specifies additional properties for this text. These properties are used as part of the algorithms in sections 2.4.6.1 (Direct Paragraph Formatting) and 2.4.6.2 (Direct Character Formatting).

**2.9.73 FcCompressed**

The FcCompressed structure specifies the location of text in the WordDocument Stream.

**fc (30 bits):** An unsigned integer that specifies an offset in the WordDocument Stream where the text starts. If fCompressed is zero, the text is an array of 16-bit Unicode characters starting at offset fc. If fCompressed is 1, the text starts at offset fc/2 and is an array of 8-bit Unicode characters, except for the values which are mapped to Unicode characters as follows.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Byte Unicode** | **Byte Unicode** | **Byte Unicode** | **Byte Unicode** | **Byte Unicode** | **Byte Unicode** |
| 0x82 0x201A | 0x86 0x2020 | 0x8A 0x0160 | 0x92 0x2019 | 0x96 0x2013 | 0x9A 0x0161 |
| 0x83 0x0192 | 0x87 0x2021 | 0x8B 0x2039 | 0x93 0x201C | 0x97 0x2014 | 0x9B 0x203A |
| 0x84 0x201E | 0x88 0x02C6 | 0x8C 0x0152 | 0x94 0x201D | 0x98 0x02DC | 0x9C 0x0153 |
| 0x85 0x2026 | 0x89 0x2030 | 0x91 0x2018 | 0x95 0x2022 | 0x99 0x2122 | 0x9F 0x0178 |

**A - fCompressed (1 bit):** A bit that specifies whether the text is compressed.

**B - r1 (1 bit):** This bit MUST be zero, and MUST be ignored.