A page addressed by one-byte codes

Every letter on the three lines below costs one byte of this file's page stream instead of two. A font is reached by glyph index unless it is told otherwise, two bytes a glyph, which puts every glyph the face holds within reach. Reached through an encoding it is addressed by one-byte codes instead: at most 255 of them, each standing for one character, and the file states which character each of them stands for.

Set through one-byte codes

HQF Development

An invoice, a payslip, a delivery note.

One byte a letter, and the file says which.

The codes it settled

The encoding is fitted to the words this page draws: the space keeps code 32, and every other character takes the lowest code free from 33 up, so nothing in the stream is a byte reading software has to escape. The first of them are H 33 · Q 34 · F 35 · D 36 · e 37 · v 38 · l 39 · o 40.

What the stream weighs

Drawn by glyph index, the three lines above cost 541 bytes of page stream; drawn through one-byte codes, 347 bytes, which is 194 less. Both figures are measured by the program that drew this page, not written in by hand.

What the single byte 32 buys

Words reading software can push apart.

Words reading software can push apart.

The two lines above hold the same characters. The second is drawn with word spacing set, which moves the pen at the single byte 32 and at no other code. Under glyph indices no code is a single byte 32, so the operator has nothing to reach and the words never move; the space of this encoding sits at 32, so they do.

One byte addresses 255 characters at a time, so a page of several alphabets is still reached by glyph index. No ligature and no small capital is drawn through an encoding either: those are glyphs no character stands for, and a code stands for a character. A file that claims archival conformance states the Windows encoding as the one its codes are read against, which this one does.