L2/03-155R

Subject: Unicode interpretation of SOFT HYPHEN breaks ISO 8859-1 compatibility

From: Markus Kuhn, Computer Laboratory, University of Cambridge,
`http://www.cl.cam.ac.uk/~mgk25/`

Date: 2003-06-04

The recent clarification of the semantics of SOFT HYPHEN (U+00AD) in Unicode 4.0 [1, 2] did not undergo any public review [3] and appears problematic with regard to existing implementation practice. Changing its general category from "Pd" (punctuation, dash) to "Cf" (other, format) along with clarifying that it has no rendering breaks backwards compatibility with how this character has been widely used in ISO 8859-1 terminals for the past 15 years. The change now causes headaches for the designers of VT100-style terminal emulators with ISO 8859-1 and UTF-8 support.

Modern text processing systems make a clear distinction between an unformatted content data stream (e.g., a Word, TeX or HTML file) and a formatted presentation datastream (e.g., a PDF, PostScript, or PCL file). This distinction is today given for granted by people who grew up with Word and similar content-data-stream editors. Unicode with its character/glyph model is clearly targeted primarily for use in content data streams, which still have to undergo line breaking before display. However, ISO 8859-1 was written in about 1984, when internationalization was all about adding diacritical characters for European countries, which preserved the one-to-one character/glyph mapping of traditional ASCII usage. At that time, in most electronic communication, content and data presentation streams were highly similar, often identical, and commonly substituted for each other. Converting presentation data streams back into plain-text content files was common practice in the mid 1980s, when ISO 8859-1 was written. It is still very common practice for any Unix users who cut&paste plain text from an `xterm` screen into a text editor, or for recipients of email with MIME content type `text/plain` [4].

As Unicode claims for U+0000 to U+00FF to be compatible with ISO 8859-1, it should also respect the intended and de-facto use of ISO 8859-1 characters and should not change their semantics over a decade later.

As discussed in more detail by Korpela [5], the ISO 8859-1 standard defines in section 6.3.3 the SOFT HYPHEN as "[a] graphic character that is imaged by a graphic symbol identical with, or similar to, that representing hyphen".

The ISO 8859-1 standard uses unfortunately only the rather unclear words "for use when a line break has been established within a word" as the complete definition of the intended usage of this character. This admittedly is hardly a clear specification and clearly falls short of setting up a document processing model and defining unambiguously what role SOFT HYPHEN plays it its various phases and functions. However, to people who routinely convert formatted presentation data streams (such as formatted ISO 8859-1 text sent to a terminal or printer) back into unformatted plain text in word processors, the meaning was obvious, as it addressed a practical need (see [5]). Hyphens that were inserted by the line-breaking algorithm at the end of a line into the presentation data stream needed to be distinguished in formatted text from hyphens that were already present in the unformatted content data stream. Only this way could they be removed automatically when converting formatted presentation data back into unformatted plaintext that will be reformatted later. Text processing practitioners are well familiar with the problems that hyphenation hyphens can show up accidentally within lines when formatted text is imported into another application.

The definition "graphic character that is imaged by a graphic symbol identical with, or similar to, that representing hyphen" made it clear to users familiar with the above mentioned problem that the SOFT HYPHEN is just an alternative of the normal graphical character HYPHEN, for use when a hyphen is inserted by a line formatting routine. For an output device such as a printer or terminal, it can then be treated exactly like the graphical character HYPHEN. It was even placed 128 code positions above the hyphen, to limit the visual damage caused when the most-significant bit in a byte was accidentally cleared during transmission.

Many years later came the HTML specification. Its authors deal exclusively with an unformatted content data stream and therefore had no need for what appears to be the original motivation for inserting the SOFT HYPHEN in about 1984 into the ANSI draft 8-bit character set proposal that later evolved into ECMA-94 and ISO 8859-1.

The original HTML 2 specification [6] by Tim Berners-Lee et al., still wisely leaves the semantics of SOFT HYPHEN untouched with the remark

```
NOTE - Use of the non-breaking space and soft hyphen indicator
characters is discouraged because support for them is not widely
deployed.
```

Unfortunately by HTML 4 [7], this had mutated into a complete reinterpretation of the purpose of the SOFT HYPHEN, compared to how it had been used over the past decade in output devices. What was originally a graphical character had turned into an invisible marker for a hyphenation opportunity:

> 9.3.3 Hyphenation
>
> In HTML, there are two types of hyphens: the plain hyphen and the soft hyphen. The plain hyphen should be interpreted by a user agent as just another character. The soft hyphen tells the user agent where a line break can occur.
>
> Those browsers that interpret soft hyphens must observe the following semantics: If a line is broken at a soft hyphen, a hyphen character must be displayed at the end of the first line. If a line is not broken at a soft hyphen, the user agent must not display a hyphen character. For operations such as searching and sorting, the soft hyphen should always be ignored.
>
> In HTML, the plain hyphen is represented by the "-" character (`&#45;` or `&#x2D;`). The soft hyphen is represented by the character entity reference `&shy;` (`&#173;` or `&#xAD;`)

This HTML 4 reinterpretation is essentially the semantics that Unicode then adopted as well.

Nevertheless, there is a vast number of VT100 terminal emulators, printers, and similar 8-bit output devices out there that treat the SOFT HYPHEN as a full graphical character, as had been suggested by ISO 8859-1 and by the old application need to distinguish between content and hyphenation hyphens in formatted presentation data streams.

A number of applications use the SOFT HYPHEN as a graphical character in presentation data streams, and that is how terminal emulators such as `xterm` as well as printing software have used it for well over a decade. A popular example is the `gnroff` command that does the formatting behind the `man` manual-display tool on Linux systems.

I have a keen interest in the design of terminal emulator applications and I maintain a definition of the POSIX function `wcwidth()` [8]. This function is used today by a number of UTF-8 terminal applications to decide, by how many character cell positions the cursor will advance if the Unicode character provided as an argument is sent to the terminal. The algorithm for generating the `wcwidth()` semantics from Unicode tables is very simple and includes the rule

```
  - Other format characters (general category code Cf in the Unicode
    database) and ZERO WIDTH SPACE (U+200B) have a column width of 0.
```

With the change of SOFT HYPHEN from general category code “Pd” to “Cf” in the Unicode 4.0 database, this now causes terminal behavior to change from `wcwidth(0x00ad)` = 1 to `wcwidth(0x00ad)` = 0. In other words, what used to be a spacing graphical character in accordance with ISO 8859-1, always advancing the cursor by one cell after printing a hyphen glyph, is now an ignorable and usually invisible format control character.

In this sense, Unicode 4.0 breaks with the well-established tradition of interpreting the SOFT HYPHEN as a graphical character in output devices.

This seems somewhat unfortunate and is now a source for confusion. If Unicode really wants to get into the business of specifying how to encode meta-information for controlling hyphenation algorithms, it should, in my opinion, engineer a complete solution:

- add a new ignorable formatting character for marking possible hyphenation points in documents, which could be called for instance HYPHENATION POINT
- add additional hyphenation control characters for use in applications where hyphenation requires a change to the spelling of a word
- add a mechanism for attaching to a plaintext file a dictionary with hyphenation exceptions that the hyphenation algorithm should apply to all occurrences of the listed word in the text
- specify detailed guidelines for how hyphenation algorithms should apply these control functions

A formatting function can then either discard a HYPHENATION POINT (if it ended up inside a formatted line), or convert it into the graphical SOFT HYPHEN character, where the hyphenation point ended up at the end of a line in the presentation data stream. This would preserve backwards compatibility with the large number of existing ISO 8859-1 output devices that treat SOFT HYPHEN as a graphical character.

Alternatively, the UTC could decide that control of a hyphenation algorithm is beyond the scope of the standard and needs to be specified in higher-level protocols, such as word processing markup languages.

In either case, the SOFT HYPHEN would have retained its historically weakly defined ISO 8859 semantics of a graphical character.

Questions:

- Is the specification of markup information for the control of hyphenation algorithms deemed to be within the scope of Unicode?
- Does the UTC have a view on what an ISO 6429 (VT100-style) terminal or printer should do when receiving a SOFT HYPHEN encoded in ISO 8859-1 or UTF-8?
- Does the UTC agree that the SOFT HYPHEN = "Cf" revision was problematic, considering that ISO 8859 defines it as a graphical character, and should this change be reverted in the next revision?
- Should ISO JTC1/SC2/WG3 be asked to remove in the next edition of ISO 8859 the definition of SOFT HYPHEN as a graphical character?

References

[1] Eric Muller: Yes, SOFT HYPHEN is a hard problem. UTC document L2/02-279, 2002-08-14. `http://www.unicode.org/L2/L2002/02279-muller.htm`

[2] Mark Davis: Property file changes for UCD 3.2.1. UTC document L2/02-267R3, 2002-08-13. `http://www.unicode.org/L2/L2002/02267r3-prop-fixes.html`

[3] Asmus Freytag: Request for public review of reclassification of hyphens. UTC document L2/02-375, 2002-10-24.

[4] R. Gellens: The `text/plain` format parameter. RFC 2646, August 1999. `http://www.ietf.org/rfc/rfc2646.txt`

[5] Jukka Korpela: Soft hyphen (SHY) – a hard problem? 1997–2002. `http://www.cs.tut.fi/~jkorpela/shy.html`

[6] T. Berners-Lee, D. Connolly: Hypertext Markup Language – 2.0. RFC 1866, November 1995. `http://www.ietf.org/rfc/rfc1866.txt`

[7] HTML 4.01 Specification, W3C Recommendation, 24 December 1999. Section 9.3.3: Hyphenation. `http://www.w3.org/TR/html4/struct/text.html#h-9.3.3`

[8] Markus Kuhn: Implementation of `wcwidth()` and `wcswidth()`. 2002-05-08. `http://www.cl.cam.ac.uk/~mgk25/ucs/wcwidth.c`