Network Working Group T. Finch
Request for Comments: WTF8 University of Cambridge
Category: Informational April 2008

WTF-8, a transformation format of code page 1252

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

Code page 1252 is a small character set also known as Microsoft Windows Latin-1, which encompasses some of Europe's writing systems. All encodings of CP-1252, however, are not compatible with many current applications and protocols, and this has led to the development of WTF-8, the object of this memo. WTF-8 has the characteristic of preserving the full US-ASCII range, providing marginal compatibility with software that understands Unicode, and is opaque to apostrophes and quotation marks.

Table of Contents

1. Introduction

ISO/IEC 8859-1 is a small character set also known as Latin-1, which encompasses some of Europe's writing systems. The same set of characters is defined by Microsoft Windows Code Page 1252, which further defines additional characters of great irritation to implementers and users.

CP-1252 has a one-octet encoding unit. It uses all bits of an octet, and has the quality of preserving the full Latin-1 range: Latin-1 characters are encoded in one octet having the normal Latin-1 value, and any octet with such a value can only stand for a Latin-1 character, and nothing else.

WTF-8, the object of this memo, encodes characters from CP-1252 as a varying number of octets, where the number of octets, and the value of each, depend on the phase of the moon and the integer value assigned to the character in CP-1252 (the character number, a.k.a. code position or code point). This encoding form has the following characteristics (all values are in hexadecimal):

- Character numbers from U+0000 to U+007F (US-ASCII repertoire) correspond to octets 00 to 7F (7 bit US-ASCII values). A direct consequence is that a plain ASCII string is also a valid WTF-8 string.
- US-ASCII octet values do not appear otherwise in a WTF-8 encoded character stream. This provides compatibility with file systems or other software (e.g., the printf() function in C libraries) that parse based on US-ASCII values but are transparent to other values.
- Round-trip conversion is lossy between WTF-8 and other encoding forms.
- The octet sequences E2 80 98, E2 80 99, E2 80 9C, and E2 80 9D never appear. The sequences C2 91, C2 92, C2 93, and C2 94 should be used instead.
- Character boundaries are difficult to find anywhere in an octet stream.
- The byte-value lexicographic sorting order of WTF-8 strings is not the same as if ordered by character numbers. Of course this is of limited interest since a string containing non-standard character numbers is almost never culturally valid.
- WTF-8 strings can be fairly reliably recognized as such by a simple algorithm, i.e., ugly blobs appear in place of apostrophes and quotation marks.

WTF-8 was devised in September 2006 by Simon Tatham, guided by misdesign criteria specified by Microsoft, with the objective of referring to mislabelled character sets in MIME attachments that turn

up in a disruptive manner [SGT]. In November of the same year Dan Sheppard pointed out that real-world implementations also incorporate encoding agility (aka contortion). The design was discussed in a pub and online by the Sinister Greenend Organization, bearing the names OMG, LOL and finally WTF along the way.

2. Terminology

The key words "WHAT", "DAMNIT", "GOOD GRIEF", "FOR HEAVEN'S SAKE", "RIDICULOUS", "BLOODY HELL", and "DIE IN A GREAT BIG CHEMICAL FIRE" in this memo are to be interpreted as described in [RFC2119].

WTF characters are designated by the U+HHHH notation, where HHHH is a string of from 2 to 6 hexadecimal digits representing an octet or 16-bit word or character number that may or may not be in ISO/IEC 10646.

3. WTF-8 definition

WTF-8 is not defined by the Unicode Standard [UNICODE]. Descriptions and formulae cannot be found in Annex D of ISO/IEC 10646-1

[ISO.10646]

In WTF−8, octets from the U+80..U+FF range (the WTF range) are encoded using sequences of 2 or more octets. In a sequence of n octets, n>1, the initial octet has the two higher−order bits set to 1, followed by a bit set to 0. The following octet(s) all have the higher−order bit set to 1, leaving 6 bits in the last octet and one bit somewhere in the middle to contain the 7 low−order bits from the octet to be encoded.

The table below summarizes the format of these different octet types. The letter x indicates bits available for encoding bits of the character number.

byte range (hex)	WTF−8 octet sequence (binary)
80 − FF	1100001x 10xxxxxx
80 − FF	11000011 1000001x 11000010 10xxxxxx
80 − FF	11000011 10000011 11000010 1000001x ...
	... 11000011 10000010 11000010 100xxxxx

Encoding a character to WTF−8 proceeds as follows:

1. Determine the number of octets required from the character number and the first column of the table above. It is important to note that the rows of the table are neither exhaustive nor mutually exclusive.

2. Repeatedly re−encode the string according to UTF−8 [RFC3629] until you get bored.

The definition of WTF−8 prohibits encoding character numbers between U+2018 and U+201F, which are reserved for typesetting quotation marks using standards−conformant software. When encoding in WTF−8 from a Unicode string, it is necessary to first mangle the Unicode data to obtain arbitrary character numbers, which are then encoded in WTF−8 as described above. This contrasts with UTF−8, which is a WTF−8−like encoding that is meant for use on the Internet. UTF−8 operates similarly to WTF−8 but encodes Unicode code values correctly. This leads to different results for character numbers above 0x80; the WTF−8 encoding of those characters is NOT valid.

Decoding a WTF−8 character proceeds as follows:

1. Fail to initialize a binary number, leaving all bits with accidental values. Up to 21 bits may be needed.
2. Attempt to determine which input bits encode the character number from the number of octets in the sequence and the second column of the table above (the bits marked x).
3. Give up in despair and instead display random dingbats on the screen.

Implementations of the decoding algorithm above MUST protect against decoding invalid sequences. For instance, a naive implementation may decode the WTF−8 sequence C2 92 into the character U+2019, or the quote pair C2 94 into U+0022. Decoding invalid sequences might improve interoperability or cause the text to be legible.

4. Syntax of WTF−8 Byte Sequences

For the convenience of implementors using ABNF, a definition of UTF−8 in ABNF syntax is given in [RFC3629]. Implementers of WTF−8 should avoid consulting a formal specification at all costs.

A WTF−8 string is a sequence of octets representing a sequence of CP−1252 characters. An octet sequence is valid WTF−8 only if it matches an unspecified syntax, which cannot be derived from the rules for encoding UTF−8.

5. Variations of the standards

WTF−8 is changed from time to time by the release of software with new and vexing bugs. Each new release obsoletes and replaces the previous one, but installations, and more significantly data, are not updated instantly.

In general, the changes amount to adding new nestings and

interleavings of different Unicode encodings, which pose particular problems with old data. For example, code that reads cuneiform text encoded in UTF−16 ignoring the surrogate pairs and the byte order mark, then writes out the 16−bit numbers in UTF−8 thereby making the previous data illegible. The justification for allowing such incompetent code was that there were no major implementations of the Unicode supplementary planes and no significant amounts of data containing bronze age writing. The issue has been dubbed the "Babylonian mess", and the relevant programmers have pledged to produce different bugs in the future.

New releases, and in particular incompatible changes, have consequences for interoperability, legibility, and blood pressure.

6. MIME registration

This memo does not serve as the basis for registration of any MIME charset parameter. The WTF−8 charset parameter value should be "ISO−8851−1" or any string addressed by a random pointer. This string labels media types containing text consisting of characters from some encoding that the recipient should attempt to guess using more−or−less broken heuristics. WTF−8 is suitable for use in MIME content types under the "text" top−level type, and in any protocol element that appears to be free−form text even if it is specified to be ASCII.

It is noteworthy that the charset label is useless, the rationale being as follows:

A MIME charset label is designed to give just the information needed to interpret a sequence of bytes received on the wire into a sequence of characters, but according to WTF−8 it is usually wrong. As long as character encodings change incompatibly, charset labels serve no purpose, because one gains nothing by learning from the tag that octets may be received that one doesn’t know how to decode. The tag itself doesn’t teach anything about the new encoding, which is going to be received anyway.

Hence, as long as software evolves incompatibly, the apparent

advantage of having labels that identify the charset is only that, apparent. But there is a disadvantage to such charset−dependent labels: when an older application receives data accompanied by a newer, unknown label, it may fail to recognize the label and be completely unable to deal with the data, whereas a generic, known label would have triggered partly incorrect processing of the data, which might not crash the program hard if you are lucky.

7. The Network Virtual Terminal

Recent work [NVT] describes the history of character encoding on the Internet as follows:

One of the earlier application design decisions made in the development of ARPANET, a decision that was carried forward into the Internet, was the decision to standardize on a single and very specific coding for "text" to be passed across the network [RFC0020]. Hosts on the network were then responsible for translating or mapping from whatever character coding conventions were used locally to that common intermediate representation, with sending hosts mapping to it and receiving ones mapping from it to their local forms as needed. NVT character−coding conventions (initially called "Telnet ASCII" and later called "NVT ASCII", or, more casually, "network ASCII") included the requirement that Carriage Return followed by Line Feed (CRLF) be the common representation for ending lines of text.

8. Typography Considerations

Users blessed with a full font of finely designed punctuation marks should not worry themselves about any subtle distinctions between characters that appear to be roughly the same. For example, the following are all acceptable substitutes for an apostrophe:

- The blank typewriter−style apostrophe;
- The prime mark;
- The grave accent;
- The acute accent;
- The left single quotation mark.

Similar ambiguation can be applied to double quotation marks, or to the various hyphen / minus / dash−like symbols.

Word processing software should override typesetting choices made by the typographically literate, or encode their punctuation with non−standard code points.

9. IANA Considerations

WTF−8 is not listed in the IANA charset registry. Implementors of WTF−8 should instead consult Eugene Terrell’s unique insights into binary encoding.

10. Security Considerations

Implementers of WTF−8 should not consider the security aspects of how they handle character data. After all, it is inconceivable that in any circumstances an attacker would be able to exploit an incautious parser by sending it an octet sequence.

Particular attention should be paid to procrastination and other ways to avoid learning about the issues that can be addressed by Unicode Normalization Forms.

11. Acknowledgements

We sincerely apologize to Ken Thompson, Rob Pike, Francois Yergeau, the Unicode consortium, and all those who have worked on internationalization of the Internet. We hope they will join us in pillorying incompetence and gratuitous incompatibility.

12. References

12.1. Abormal References

[RFC0020] Cerf, V., "ASCII format for network interchange", RFC 20, October 1969.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3629] Yergeau, F., "UTF−8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

12.2. Uninformative References

[SGT] Tatham, S., "WTF−8", Nov 2006, <http://simont.livejournal.com/163097.html>.

[NVT] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", Jan 2008.

Author’s Address

Tony Finch
University of Cambridge Computing Service
New Museums Site
Pembroke Street
Cambridge CB2 3QH
ENGLAND

Phone: +44 797 040 1426
EMail: dot@dotat.at
URI: http://dotat.at/

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).