Network Working Group
Request for Comments: 2044
Category: Informational

F. Yergeau
Alis Technologies

October 1996

UTF-8, a transformation format of Unicode and ISO 10646

Status of this Memo

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

Abstract

The Unicode Standard, version 1.1, and ISO/IEC 10646-1:1993 jointly define a 16 bit character set which encompasses most of the world's writing systems. 16-bit characters, however, are not compatible with many current applications and protocols, and this has led to the development of a few so-called UCS transformation formats (UTF), each with different characteristics. UTF-8, the object of this memo, has the characteristic of preserving the full US-ASCII range: US-ASCII characters are encoded in one octet having the usual US-ASCII value, and any octet with such a value can only be an US-ASCII character. This provides compatibility with file systems, parsers and other software that rely on US-ASCII values but are transparent to other values.

1. Introduction

The Unicode Standard, version 1.1 [UNICODE], and ISO/IEC 10646-1:1993 [ISO-10646] jointly define a 16 bit character set, UCS-2, which encompasses most of the world's writing systems. ISO 10646 further defines a 31-bit character set, UCS-4, with currently no assignments outside of the region corresponding to UCS-2 (the Basic Multilingual Plane, BMP). The UCS-2 and UCS-4 encodings, however, are hard to use in many current applications and protocols that assume 8 or even 7 bit characters. Even newer systems able to deal with 16 bit characters cannot process UCS-4 data. This situation has led to the development of so-called UCS transformation formats (UTF), each with different characteristics.

UTF-1 has only historical interest, having been removed from ISO 10646. UTF-7 has the quality of encoding the full Unicode repertoire using only octets with the high-order bit clear (7 bit US-ASCII values, [US-ASCII]), and is thus deemed a mail-safe encoding ([RFC1642]). UTF-8, the object of this memo, uses all bits of an octet, but has the quality of preserving the full US-ASCII range:

US-ASCII characters are encoded in one octet having the normal US-ASCII value, and any octet with such a value can only stand for an US-ASCII character, and nothing else.

UTF-16 is a scheme for transforming a subset of the UCS-4 repertoire into a pair of UCS-2 values from a reserved range. UTF-16 impacts UTF-8 in that UCS-2 values from the reserved range must be treated specially in the UTF-8 transformation.

UTF-8 encodes UCS-2 or UCS-4 characters as a varying number of octets, where the number of octets, and the value of each, depend on the integer value assigned to the character in ISO 10646. This transformation format has the following characteristics (all values are in hexadecimal):

- Character values from 0000 0000 to 0000 007F (US-ASCII repertoire) correspond to octets 00 to 7F (7 bit US-ASCII values).

- US-ASCII values do not appear otherwise in a UTF-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 easy between UTF-8 and either of UCS-4, UCS-2 or Unicode.

- The first octet of a multi-octet sequence indicates the number of octets in the sequence.

- Character boundaries are easily found from anywhere in an octet stream.

- The lexicographic sorting order of UCS-4 strings is preserved. Of course this is of limited interest since the sort order is not culturally valid in either case.

- The octet values FE and FF never appear.

UTF-8 was originally a project of the X/Open Joint Internationalization Group XOJIG with the objective to specify a File System Safe UCS Transformation Format [FSS-UTF] that is compatible with UNIX systems, supporting multilingual text in a single encoding. The original authors were Gary Miller, Greger Leijonhufvud and John Entenmann. Later, Ken Thompson and Rob Pike did significant work for the formal UTF-8.

A description can also be found in Unicode Technical Report #4 [UNICODE]. The definitive reference, including provisions for UTF-16 data within UTF-8, is Annex R of ISO/IEC 10646-1 [ISO-10646].

2. UTF-8 definition

In UTF-8, characters are encoded using sequences of 1 to 6 octets. The only octet of a "sequence" of one has the higher-order bit set to 0, the remaining 7 bits being used to encode the character value. In a sequence of n octets, n>1, the initial octet has the n higher-order bits set to 1, followed by a bit set to 0. The remaining bit(s) of that octet contain bits from the value of the character to be encoded. The following octet(s) all have the higher-order bit set to 1 and the following bit set to 0, leaving 6 bits in each to contain bits from the character 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 UCS-4 character value.

UCS-4 range (hex.)	UTF-8 octet sequence (binary)
0000 0000-0000 007F	0xxxxxxx
0000 0080-0000 07FF	110xxxxx 10xxxxxx
0000 0800-0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx
0001 0000-001F FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
0020 0000-03FF FFFF	111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
0400 0000-7FFF FFFF	1111110x 10xxxxxx ... 10xxxxxx

Encoding from UCS-4 to UTF-8 proceeds as follows:

1) Determine the number of octets required from the character value and the first column of the table above.

2) Prepare the high-order bits of the octets as per the second column of the table.

3) Fill in the bits marked x from the bits of the character value, starting from the lower-order bits of the character value and putting them first in the last octet of the sequence, then the next to last, etc. until all x bits are filled in.

The algorithm for encoding UCS-2 (or Unicode) to UTF-8 can be obtained from the above, in principle, by simply extending each UCS-2 character with two zero-valued octets. However, UCS-2 values between D800 and DFFF, being actually UCS-4 characters transformed through UTF-16, need special treatment: the UTF-16 transformation must be undone, yielding a UCS-4 character that is then transformed as above.

Decoding from UTF-8 to UCS-4 proceeds as follows:

1) Initialize the 4 octets of the UCS-4 character with all bits set to 0.

2) Determine which bits encode the character value from the number of octets in the sequence and the second column of the table above (the bits marked x).

3) Distribute the bits from the sequence to the UCS-4 character, first the lower-order bits from the last octet of the sequence and proceeding to the left until no x bits are left.

   If the UTF-8 sequence is no more than three octets long, decoding can proceed directly to UCS-2 (or equivalently Unicode).

   A more detailed algorithm and formulae can be found in [FSS_UTF], [UNICODE] or Annex R to [ISO-10646].

3. Examples

The Unicode sequence "A<NOT IDENTICAL TO><ALPHA>." (0041, 2262, 0391, 002E) may be encoded as follows:

```
41 E2 89 A2 CE 91 2E
```

The Unicode sequence "Hi Mom <WHITE SMILING FACE>!" (0048, 0069, 0020, 004D, 006F, 006D, 0020, 263A, 0021) may be encoded as follows:

```
48 69 20 4D 6F 6D 20 E2 98 BA 21
```

The Unicode sequence representing the Han characters for the Japanese word "nihongo" (65E5, 672C, 8A9E) may be encoded as follows:

```
E6 97 A5 E6 9C AC E8 AA 9E
```

MIME registrations

This memo is meant to serve as the basis for registration of a MIME character encoding (charset) as per [RFC1521]. The proposed charset parameter value is "UTF-8". This string would label media types containing text consisting of characters from the repertoire of ISO 10646-1 encoded to a sequence of octets using the encoding scheme outlined above.

Security Considerations

Security issues are not discussed in this memo.

Acknowledgments

The following have participated in the drafting and discussion of this memo:

James E. Agenbroad	Andries Brouwer
Martin J. D|rst	David Goldsmith
Edwin F. Hart	Kent Karlsson
Markus Kuhn	Michael Kung
Alain LaBonte	Murray Sargent
Keld Simonsen	Arnold Winkler

Bibliography

[FSS_UTF] X/Open CAE Specification C501 ISBN 1-85912-082-2 28cm. 22p. pbk. 172g. 4/95, X/Open Company Ltd., "File System Safe UCS Transformation Format (FSS_UTF)", X/Open Preleminary Specification, Document Number P316. Also published in Unicode Technical Report #4.

[ISO-10646] ISO/IEC 10646-1:1993. International Standard -- Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane. UTF-8 is described in Annex R, adopted but not yet published. UTF-16 is described in Annex Q, adopted but not yet published.

[RFC1521] Borenstein, N., and N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 1521, Bellcore, Innosoft, September 1993.

[RFC1641] Goldsmith, D., and M. Davis, "Using Unicode with MIME", RFC 1641, Taligent inc., July 1994.

[RFC1642] Goldsmith, D., and M. Davis, "UTF-7: A Mail-safe Transformation Format of Unicode", RFC 1642, Taligent, Inc., July 1994.

[UNICODE] The Unicode Consortium, "The Unicode Standard -- Worldwide Character Encoding -- Version 1.0", Addison-Wesley, Volume 1, 1991, Volume 2, 1992. UTF-8 is described in Unicode Technical Report #4.

[US-ASCII] Coded Character Set--7-bit American Standard Code for Information Interchange, ANSI X3.4-1986.

Author's Address

Francois Yergeau
Alis Technologies
100, boul. Alexis-Nihon
Suite 600
Montreal QC H4M 2P2
Canada

Tel: +1 (514) 747-2547
Fax: +1 (514) 747-2561
EMail: fyergeau@alis.com