INTERNET DRAFT
Expires: 23 June 1998
<draft-abela-utf9-00.txt>

J. Abela
HSC

23 December 1997

UTF-9, a transformation format of UCS

Status of this Memo

Abstract

ISO/IEC 10646 defines a multi-octet character set called the Universal Character Set (UCS) which encompasses most of the world's writing systems. Multi-octet 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-9, the object of this memo, has the characteristic of preserving the full ISO-Latin1 range, providing compatibility with file systems, parsers and other software that rely on ISO-Latin1 values.

ISO-Latin1 is almost as widespread as ASCII in many countries, especially in most of western Europe, and is the default character set for HTML. A compatible encoding seems desirable, where possible.

1. Introduction

ISO/IEC 10646-1 [ISO-10646] defines a multi-octet character set called the Universal Character Set (UCS), which encompasses most of the world's writing systems. Two multi-octet encodings are defined, a four-octet per character encoding called UCS-4 and a two-octet per character encoding called UCS-2, able to address only the first 64K characters of the UCS (the Basic Multilingual Plane, BMP), outside of which there are currently no assignments.

It is noteworthy that the same set of characters is defined by the Unicode standard [UNICODE], which further defines additional character properties and other application details of great interest to implementors, but does not have the UCS-4 encoding. Up to the present time, changes in Unicode and amendments to ISO/IEC 10646 have tracked each other, so that the character repertoires and code point assignments have remained in sync. The relevant standardization committees have committed to maintain this very useful synchronism.

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/IEC 10646. UTF-7 has the quality of encoding the full BMP 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 ([RFC2152]). UTF-8 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-9, the object of this memo, has the quality of preserving the full ISO-Latin1 range: ISO-Latin1 characters are encoded in one octet having the normal ISO-Latin1 value.

UTF-16 is a scheme for transforming a subset of the UCS-4 repertoire into pairs of UCS-2 values from a reserved range. UTF-16 impacts UTF-9 in that UCS-2 values from the reserved range must be treated specially in the UTF-9 transformation.

UTF-9 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/IEC 10646. This transformation format has the following characteristics (all values are in hexadecimal):

- Character values from 0000 0000 to 0000 007F and 0000 00A0 to 0000 00FF (Latin1 repertoire) correspond to octets 00 to 7F and A0 to FF (8 bit Latin1 values). A direct consequence is that a plain Latin1 string is also a valid UTF-9 string. Note that Latin1 octets in a UTF-9 string may be non-Latin1 characters.

- US-ASCII values do not appear otherwise in a UTF-9 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. However, note that Latin1 octets in a UTF-9 stream may be non-Latin1

characters when used as part of multi-octet sequences.

- Round-trip conversion is easy between UTF-9 and either of UCS-4, UCS-2.

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

- UTF-9 encoding length is never bigger than UTF-8.

- unlike UTF-8, there is no reliable way to find character boundaries in a UTF-9 octet stream.

UTF-9 is heavily based on UTF-8 definition. More information about UTF, Unicode, and their various versions can be found in RFC-2044.

UTF-9 definition

In UTF-9, characters are encoded using sequences of 1 to 5 octets. The only octet of a "sequence" of one is in the ranges 00 to 7F or A0-FF. In a sequence of n octets, n>1, the initial octet is in the range 80 to 9F. This octet specifies the length of the sequence and contains value bits if in the range 80 to 8F. All the bits of the remaining octets are used to encode the character.

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-9 octet sequence (binary)
0000 0000-0000 007F	0xxxxxxx
0000 00A0-0000 00BF	101xxxxx
0000 00C0-0000 00FF	11xxxxxx
0000 0100-0000 07FF	1000xxxx 1xxxxxxx
0000 0800-0000 FFFF	100100xx 1xxxxxxx 1xxxxxxx
0001 0000-007F FFFF	100101xx 1xxxxxxx 1xxxxxxx 1xxxxxxx
0080 0000-7FFF FFFF	10011xxx 1xxxxxxx 1xxxxxxx 1xxxxxxx 1xxxxxxx

Examples

The Latin1 sequence "No<e diaeresis>l" should be encoded as follows:

```
UCS-2: 004E 006F 00EB 006C
UTF-9: 4E   6F   EB   6C
UTF-8: 4E   6F   C3AB 6C
```

The UCS-2 sequence "A<NOT IDENTICAL TO><ALPHA>." should be encoded as follows:

```
UCS-2: 0041  2262      0391   002E
UTF-9: 41    90 C4 E2  87 91  2E
UTF-8: 41    E2 89 A2  CE 91  2E
```

The UCS-2 sequence representing the Hangul characters for the Korean word "hangugo" should be encoded as follows:

```
 UCS-2: D55C      AD6D      C5B4
 UTF-9: 93 AA DC  92 DA ED  93 8B B4
 UTF-8: ED 95 9C  EA B5 AD  EC 96 B4
```

Security Considerations

Implementors of UTF-9 need to consider the security aspects of how they handle illegal UTF-9 sequences. It is conceivable that in some circumstances an attacker would be able to exploit an incautious UTF-9 parser by sending it an octet sequence that is not permitted by the UTF-9 syntax.

A particularly subtle form of this attack could be carried out against a parser which performs security-critical validity checks against the UTF-9 encoded form of its input, but interprets certain illegal octet sequences as characters. For example, a parser might prohibit the NUL character when encoded as the single-octet sequence 00, but allow the illegal two-octet sequence 80 80 and interpret it as a NUL character. Another example might be a parser which prohibits the octet sequence 2F 2E 2E 2F ("/../"), yet permits the illegal octet sequence 2F 2E 80 AE 2F.

Acknowledgments

Most of the text of this memo comes from the UTF-8 memo from Francois Yergeau. The following have participated in the drafting of this memo: Antoine Leca and Francois Yergeau

Bibliography

[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. Five amendments and a technical corrigendum have been published up to now.

[RFC2152] D. Goldsmith, M. Davis, "UTF-7: A Mail-safe Transformation Format of Unicode", RFC 1642, Taligent inc., May 1997. (Obsoletes RFC1642)

[UNICODE] The Unicode Consortium, "The Unicode Standard -- Version 2.0", Addison-Wesley, 1996.

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

Author's Address

Jerome Abela
Herve Schauer Consultants
142, rue de Rivoli
75001 Paris
France

Phone: +33 141 409 700
Fax: +33 141 409 709

EMail: Jerome.Abela@hsc.fr