Network Working Group
Request for Comments: 1642
Category: Experimental

D. Goldsmith
M. Davis
Taligent, Inc.

July 1994

UTF-7

A Mail-Safe Transformation Format of Unicode

Status of this Memo

This memo defines an Experimental Protocol 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(E) jointly define a 16 bit character set (hereafter referred to as Unicode) which encompasses most of the world's writing systems. However, Internet mail (STD 11, RFC 822) currently supports only 7-bit US ASCII as a character set. MIME (RFC 1521 and RFC 1522) extends Internet mail to support different media types and character sets, and thus could support Unicode in mail messages. MIME neither defines Unicode as a permitted character set nor specifies how it would be encoded, although it does provide for the registration of additional character sets over time.

This document describes a new transformation format of Unicode that contains only 7-bit ASCII characters and is intended to be readable by humans in the limiting case that the document consists of characters from the US-ASCII repertoire. It also specifies how this transformation format is used in the context of RFC 1521, RFC 1522, and the document "Using Unicode with MIME".

Motivation

Although other transformation formats of Unicode exist and could conceivably be used in this context (most notably UTF-1 and UTF-8, also known as UTF-2 or UTF-FSS), they suffer the disadvantage that they use octets in the range decimal 128 through 255 to encode Unicode characters outside the US-ASCII range. Thus, in the context of mail, those octets must themselves be encoded. This requires putting text through two successive encoding processes, and leads to a significant expansion of characters outside the US-ASCII range, putting non-English speakers at a disadvantage. For example, using UTF-FSS together with the Quoted-Printable content transfer encoding of MIME represents US-ASCII characters in one octet, but other characters may require up to nine octets.

Overview

UTF-7 encodes Unicode characters as US-ASCII, together with shift sequences to encode characters outside that range. For this purpose, one of the characters in the US-ASCII repertoire is reserved for use as a shift character.

Many mail gateways and systems cannot handle the entire US-ASCII character set (those based on EBCDIC, for example), and so UTF-7 contains provisions for encoding characters within US-ASCII in a way that all mail systems can accomodate.

UTF-7 should normally be used only in the context of 7 bit transports, such as mail and news. In other contexts, straight Unicode or UTF-8 is preferred.

See the document "Using Unicode with MIME" for the overall specification on usage of Unicode transformation formats with MIME.

Definitions

First, the definition of Unicode:

The 16 bit character set Unicode is defined by "The Unicode Standard, Version 1.1". This character set is identical with the character repertoire and coding of the international standard ISO/IEC 10646-1:1993(E); Coded Representation Form=UCS-2; Subset=300; Implementation Level=3.

Note. Unicode 1.1 further specifies the use and interaction of these character codes beyond the ISO standard. However, any valid 10646 BMP (Basic Multilingual Plane) sequence is a valid Unicode sequence, and *vice versa*; Unicode supplies interpretations of sequences on which the ISO standard is silent as to interpretation.

Next, some handy definitions of US-ASCII character subsets:

Set D (directly encoded characters) consists of the following characters (derived from RFC 1521, Appendix B): the upper and lower case letters A through Z and a through z, the 10 digits 0-9, and the following nine special characters (note that "+" and "=" are omitted):

Character	ASCII & Unicode Value (decimal)
'	39
(	40
)	41
,	44
-	45
.	46
/	47
:	58
?	63

Set O (optional direct characters) consists of the following characters (note that “\” and “~” are omitted):

Character	ASCII & Unicode Value (decimal)
!	33
"	34
#	35
$	36
%	37
&	38
*	42
;	59
<	60
=	61
>	62
@	64
[	91
]	93
^	94
_	95
`	96
{	123
|	124
}	125

Rationale. The characters “\” and “~” are omitted because they are often redefined in variants of ASCII.

Set B (Modified Base 64) is the set of characters in the Base64 alphabet defined in RFC 1521, excluding the pad character “=” (decimal value 61).

Rationale. The pad character “=” is excluded because UTF-7 is designed for use within header fields as set forth in RFC 1522. Since the only readable encoding in RFC 1522 is “Q” (based on RFC 1521’s Quoted-Printable), the “=” character is not available for use (without a lot of escape sequences). This was very unfortunate but unavoidable.The “=” character could otherwise have been used as the UTF-7 escape character as well (rather than using “+”).

Note that all characters in US-ASCII have the same value in Unicode when zero-extended to 16 bits.

UTF-7 Definition

A UTF-7 stream represents 16-bit Unicode characters in 7-bit US-ASCII as follows:

Rule 1: (direct encoding) Unicode characters in set D above may be encoded directly as their ASCII equivalents. Unicode characters in Set O may optionally be encoded directly as their ASCII equivalents, bearing in mind that many of these characters are illegal in header fields, or may not pass correctly through some mail gateways.

Rule 2: (Unicode shifted encoding) Any Unicode character sequence may be encoded using a sequence of characters in set B, when preceded by the shift character “+” (US-ASCII character value decimal 43). The “+” signals that subsequent octets are to be interpreted as elements of the Modified Base64 alphabet until a character not in that alphabet is encountered. Such characters include control characters such as carriage returns and line feeds; thus, a Unicode shifted sequence always terminates at the end of a line. As a special case, if the sequence terminates with the character “−” (US-ASCII decimal 45) then that character is absorbed; other terminating characters are not absorbed and are processed normally.

Rationale. A terminating character is necessary for cases where the next character after the Modified Base64 sequence is part of character set B. It can also enhance readability by delimiting encoded sequences.

Also as a special case, the sequence “+−” may be used to encode the character “+”. A “+” character followed immediately by any character other than members of set B or “−” is an ill-formed sequence.

Unicode is encoded using Modified Base64 by first converting Unicode 16-bit quantities to an octet stream (with the most significant octet first). Text with an odd number of octets is ill-formed.

Rationale. ISO/IEC 10646-1:1993(E) specifies that when characters in the UCS-2 form are serialized as octets, that the most significant octet appear first. This is also in keeping with common network practice of choosing a canonical format for transmission.

Next, the octet stream is encoded by applying the Base64 content transfer encoding algorithm as defined in RFC 1521, modified to omit the "=" pad character. Instead, when encoding, zero bits are added to pad to a Base64 character boundary. When decoding, any bits at the end of the Modified Base64 sequence that do not constitute a complete 16-bit Unicode character are discarded. If such discarded bits are non-zero the sequence is ill-formed.

Rationale. The pad character "=" is not used when encoding Modified Base64 because of the conflict with its use as an escape character for the Q content transfer encoding in RFC 1522 header fields, as mentioned above.

Rule 3: The space (decimal 32), tab (decimal 9), carriage return (decimal 13), and line feed (decimal 10) characters may be directly represented by their ASCII equivalents. However, note that MIME content transfer encodings have rules concerning the use of such characters. Usage that does not conform to the restrictions of RFC 822, for example, would have to be encoded using MIME content transfer encodings other than 7bit or 8bit, such as quoted-printable, binary, or base64.

Given this set of rules, Unicode characters which may be encoded via rules 1 or 3 take one octet per character, and other Unicode characters are encoded on average with 2 2/3 octets per character plus one octet to switch into Modified Base64 and an optional octet to switch out.

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

```
A+ImIDkQ.
```

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

```
Hi Mom +Jjo-!
```

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

```
+ZeVnLIqe-
```

Use of Character Set UTF-7 Within MIME

Character set UTF-7 is safe for mail transmission and therefore may be used with any content transfer encoding in MIME (except where line length and line break restrictions are violated). Specifically, the 7 bit encoding for bodies and the Q encoding for headers are both acceptable. The MIME character set identifier is UNICODE-1-1-UTF-7.

Example. Here is a text portion of a MIME message containing the Unicode sequence “Hi Mom <WHITE SMILING FACE>!” (hexadecimal 0048, 0069, 0020, 004D, 006F, 004D, 0020, 263A, 0021).

```
Content-Type: text/plain; charset=UNICODE-1-1-UTF-7

Hi Mom +Jjo-!
```

Example. Here is a text portion of a MIME message containing the Unicode sequence representing the Han characters for the Japanese word “nihongo” (hexadecimal 65E5,672C,8A9E).

```
Content-Type: text/plain; charset=UNICODE-1-1-UTF-7

+ZeVnLIqe-
```

Example. Here is a text portion of a MIME message containing the Unicode sequence “A<NOT IDENTICAL TO><ALPHA>.” (hexadecimal 0041,2262,0391,002E).

```
Content-Type: text/plain; charset=UNICODE-1-1-UTF-7

A+ImIDkQ.
```

Example. Here is a text portion of a MIME message containing the Unicode sequence “Item 3 is <POUND SIGN>1.” (hexadecimal 0049, 0074, 0065, 006D, 0020, 0033, 0020, 0069, 0073, 0020, 00A3, 0031, 002E).

```
Content-Type: text/plain; charset=UNICODE-1-1-UTF-7

Item 3 is +AKM-1.
```

Note that to achieve the best interoperability with systems that may not support Unicode or MIME, when preparing text for mail transmission line breaks should follow Internet conventions. This means that lines should be short and terminated with the proper SMTP CRLF sequence. Unicode LINE SEPARATOR (hexadecimal 2028) and PARAGRAPH SEPARATOR (hexadecimal 2029) should be converted to SMTP line breaks. Ideally, this would be handled transparently by a Unicode-aware user agent.

This preparation is not absolutely necessary, since UTF-7 and the appropriate MIME content transfer encoding can handle text that does not follow Internet conventions, but readability by systems without Unicode or MIME will be impaired. See RFC 1521 for an in-depth discussion of mail interoperability issues.

Lines should never be broken in the middle of a UTF-7 shifted sequence, since such sequences may not cross line breaks. Therefore, UTF-7 encoding should take place after line breaking. If a line containing a shifted sequence is too long after encoding, a MIME content transfer encoding such as Quoted Printable can be used to encode the text. Another possibility is to perform line breaking and UTF-7 encoding at the same time, so that lines containing shifted sequences already conform to length restrictions.

Discussion

In this section we will motivate the introduction of UTF-7 as opposed to the alternative of using the existing transformation formats of Unicode (e.g., UTF-8) with MIME's content transfer encodings. Before discussing this, it will be useful to list some assumptions about character frequency within typical natural language text strings that we use to estimate typical storage requirements:

1. Most Western European languages use roughly 7/8 of their letters from US-ASCII and 1/8 from Latin 1 (ISO-8859-1).

2. Most non-European alphabet-based languages (e.g., Greek) use about 1/6 of their letters from ASCII (since white space is in the 7-bit area) and the rest from their alphabets.

3. East Asian ideographic-based languages (including Japanese) use essentially all of their characters from the Han or CJK syllabary area.

4. Non-directly encoded punctuation characters do not occur frequently enough to affect the results.

Notice that current 8 bit standards, such as ISO-8859-x, require use of a content transfer encoding. For comparison with the subsequent discussion, the costs break down as follows (note that many of these figures are approximate since they depend on the exact composition of the text):

8859-x in Base64

Text type	Average octets/character
All	1.33

8859-x in Quoted Printable

Text type	Average octets/character
US-ASCII	1
Western European	1.25
Other	2.67

Note also that Unicode encoded in Base64 takes a constant 2.67 octets per character. For purposes of comparison, we will look at UTF-8 in Base64 and Quoted Printable, and UTF-7. UTF-1 gives results substantially similar to UTF-8. Also note that fixed overhead for long strings is relative to 1/n, where n is the encoded string length in octets.

UTF-8 in Base64

Text type	Average octets/character
US-ASCII	1.33
Western European	1.5
Some Alphabetics	2.44
All others	4

UTF-8 in Quoted Printable

Text type	Average octets/character
US-ASCII	1
Western European	1.63
Some Alphabetics	5.17
All others	7-9

UTF-7

Text type	Average octets/character
Most US-ASCII	1
Western European	1.5
All others	2.67+2/n

We feel that the UTF-8 in Quoted Printable option is not viable due to the very large expansion of all text except Western European. This would only be viable in texts consisting of large expanses of US-ASCII or Latin characters with occasional other characters interspersed. We would prefer to introduce one encoding that works reasonably well for all users.

We also feel that UTF-8 in Base64 has high expansion for non-Western-European users, and is less desirable because it cannot be read directly, even when the content is largely US-ASCII. The base encoding of UTF-7 gives competitive results and is readable for ASCII text.

UTF-7 gives results competitive with ISO-8859-x, with access to all of the Unicode character set. We believe this justifies the introduction of a new transformation format of Unicode.

As an alternative to use of UTF-7, it is possible to intermix Unicode characters with other character sets using an existing MIME mechanism, the multipart/mixed content type (thanks to Nathaniel Borenstein for pointing this out). For instance (repeating an earlier example):

```
Content-type: multipart/mixed; boundary=foo

--foo
Content-type: text/plain; charset=us-ascii

Hi Mom
--foo
Content-type: text/plain; charset=UNICODE-1-1
Content-transfer-encoding: base64

Jjo=
--foo
Content-type: text/plain; charset=us-ascii

!
--foo--
```

Theoretically, this removes the need for UTF-7 in message bodies (multipart may not be used in header fields). However, we feel that as use of the Unicode character set becomes more widespread, intermittent use of specialized Unicode characters (such as dingbats and mathematical symbols) will occur, and that text will also typically include small snippets from other scripts, such as Cyrillic, Greek, or East Asian languages (anything in the Roman script is already handled adequately by existing MIME character sets). Although the multipart technique works well for large chunks of text in alternating character sets, we feel it does not adequately support the kinds of uses just discussed, and so we still believe the introduction of UTF-7 is justified.

Summary

The UTF-7 encoding allows Unicode characters to be encoded within the US-ASCII 7 bit character set. It is most effective for Unicode sequences which contain relatively long strings of US-ASCII characters interspersed with either single Unicode characters or strings of Unicode characters, as it allows the US-ASCII portions to be read on systems without direct Unicode support.

UTF-7 should only be used with 7 bit transports such as mail and news. In other contexts, use of straight Unicode or UTF-8 is preferred.

Acknowledgements

Many thanks to the following people for their contributions, comments, and suggestions. If we have omitted anyone it was through oversight and not intentionally.

Glenn Adams
Harald T. Alvestrand
Nathaniel Borenstein
Lee Collins
Jim Conklin
Dave Crocker
Steve Dorner
Dana S. Emery
Ned Freed
Kari E. Hurtta
John H. Jenkins
John C. Klensin
Valdis Kletnieks
Keith Moore
Masataka Ohta
Einar Stefferud
Erik M. van der Poel

Appendix A -- Examples

Here is a longer example, taken from a document originally in Big5 code. It has been condensed for brevity. There are two versions: the first uses optional characters from set O (and thus may not pass through some mail gateways), and the second uses no optional characters.

```
Content-type: text/plain; charset=unicode-1-1-utf-7

Below is the full Chinese text of the Analects (+itaKng-).

The sources for the text are:

"The sayings of Confucius," James R. Ware, trans.  +U/BTFw-:
+ZYeB9FH6ckh5Pg-, 1980.  (Chinese text with English translation)

+Vttm+E6UfZM-, +W4tRQ066bOg-, +UxdOrA-:  +Ti1XC2b4Xpc-, 1990.

"The Chinese Classics with a Translation, Critical and
Exegetical Notes, Prolegomena, and Copius Indexes," James
Legge, trans., Taipei:  Southern Materials Center Publishing,
Inc., 1991.  (Chinese text with English translation)

Big Five and GB versions of the text are being made available
separately.

Neither the Big Five nor GB contain all the characters used in
this text.  Missing characters have been indicated using their
Unicode/ISO 10646 code points.  "U+-" followed by four
hexadecimal digits indicates a Unicode/10646 code (e.g.,
U+-9F08).  There is no good solution to the problem of the small
size of the Big Five/GB character sets; this represents the
solution I find personally most satisfactory.

(omitted...)

I have tried to minimize this problem by using variant
characters where they were available and the character
actually in the text was not.  Only variants listed as such in
the +XrdxmVtXUXg- were used.

(omitted...)

John H. Jenkins
+TpVPXGBG-
John_Jenkins@taligent.com
5 January 1993
(omitted...)
```

```
Content-type: text/plain; charset=unicode-1-1-utf-7

Below is the full Chinese text of the Analects (+itaKng-).

The sources for the text are:

+ACI-The sayings of Confucius,+ACI- James R. Ware, trans.  +U/BTFw-:
+ZYeB9FH6ckh5Pg-, 1980.  (Chinese text with English translation)

+Vttm+E6UfZM-, +W4tRQ066bOg-, +UxdOrA-:  +Ti1XC2b4Xpc-, 1990.

+ACI-The Chinese Classics with a Translation, Critical and
Exegetical Notes, Prolegomena, and Copius Indexes,+ACI- James
Legge, trans., Taipei:  Southern Materials Center Publishing,
Inc., 1991.  (Chinese text with English translation)

Big Five and GB versions of the text are being made available
separately.

Neither the Big Five nor GB contain all the characters used in
this text.  Missing characters have been indicated using their
Unicode/ISO 10646 code points.  +ACI-U+-+ACI- followed by four
hexadecimal digits indicates a Unicode/10646 code (e.g.,
U+-9F08).  There is no good solution to the problem of the small
size of the Big Five/GB character sets+ADs- this represents the
solution I find personally most satisfactory.

(omitted...)

I have tried to minimize this problem by using variant
characters where they were available and the character
actually in the text was not.  Only variants listed as such in
the +XrdxmVtXUXg- were used.

(omitted...)

John H. Jenkins
+TpVPXGBG-
John+AF8-Jenkins+AEA-taligent.com
5 January 1993
(omitted...)
```

Security Considerations

Security issues are not discussed in this memo.

References

[UNICODE 1.1] "The Unicode Standard, Version 1.1": Version 1.0, Volume 1 (ISBN 0-201-56788-1), Version 1.0, Volume 2 (ISBN 0-201-60845-6), and "Unicode Technical Report #4, The Unicode Standard, Version 1.1" (available from The Unicode Consortium, and soon to be published by Addison-Wesley).

[ISO 10646] ISO/IEC 10646-1:1993(E) Information Technology--Universal Multiple-octet Coded Character Set (UCS).

[MIME/UNICODE] Goldsmith, D., and M. Davis, "Using Unicode with MIME", RFC 1641, Taligent, Inc., July 1994.

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

[ISO-8859] Information Processing -- 8-bit Single-Byte Coded Graphic Character Sets -- Part 1: Latin Alphabet No. 1, ISO 8859-1:1987. Part 2: Latin alphabet No. 2, ISO 8859-2, 1987. Part 3: Latin alphabet No. 3, ISO 8859-3, 1988. Part 4: Latin alphabet No. 4, ISO 8859-4, 1988. Part 5: Latin/Cyrillic alphabet, ISO 8859-5, 1988. Part 6: Latin/Arabic alphabet, ISO 8859-6, 1987. Part 7: Latin/Greek alphabet, ISO 8859-7, 1987. Part 8: Latin/Hebrew alphabet, ISO 8859-8, 1988. Part 9: Latin alphabet No. 5, ISO 8859-9, 1990.

[RFC822] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, UDEL, August 1982.

[RFC-1521] 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.

[RFC-1522] Moore, K., "Representation of Non-Ascii Text in Internet Message Headers" RFC 1522, University of Tennessee, September 1993.

[UTF-8] X/Open Company Ltd., "File System Safe UCS Transformation Format (FSS_UTF)", X/Open Preliminary Specification, Document Number: P316. This information also appears in Unicode Technical Report #4, and in a forthcoming annex to ISO/IEC 10646.

Authors' Addresses

David Goldsmith
Taligent, Inc.
10201 N. DeAnza Blvd.
Cupertino, CA 95014-2233

Phone: 408-777-5225
Fax: 408-777-5081
EMail: david_goldsmith@taligent.com

Mark Davis
Taligent, Inc.
10201 N. DeAnza Blvd.
Cupertino, CA 95014-2233

Phone: 408-777-5116
Fax: 408-777-5081
EMail: mark_davis@taligent.com