Internet Engineering Task Force (IETF)
INTERNET-DRAFT
draft-ietf-idn-utf6-00
November 16, 2000

Mark Welter
Brian W. Spolarich
WALID, Inc.

Expires May 16, 2001

UTF-6 - Yet Another ASCII-Compatible Encoding for IDN

Status of this memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

> The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt
>
> The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

The distribution of this document is unlimited.

Abstract

This document describes a tranformation method for representing Unicode character codepoints in host name parts in a fashion that is completely compatible with the current Domain Name System. It is proposed as a potential candidate for an ASCII-Compatible Encoding (ACE) for supporting the deployment of an internationalized Domain Name System. The tranformation method, an extension of the UTF-5 encoding proposed by Duerst, provides both for more efficient representation of typical Unicode sequences while preserving simplicity and readability. This transformation method is deployed as part of the current WALID multilingual domain name system implementation, although that status should not necessarily influence the evaluation of its merits as a candidate encoding method.

Table of Contents

1. Introduction

UTF-6 describes an encoding scheme of the ISO/IEC 10646 [ISO10646] character set (whose character code assignments are synchronized with Unicode [UNICODE3]), and the procedures for using this scheme to transform host name parts containing Unicode character sequences into sequences that are compatible with the current DNS protocol [STD13]. As such, it satisfies the definition of a 'charset' as defined in [IDNREQ].

1.1 Terminology

The key words "MUST", "SHALL", "REQUIRED", "SHOULD", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Hexadecimal values are shown preceded with an "0x". For example, "0xa1b5" indicates two octets, 0xa1 followed by 0xb5. Binary values are shown preceded with an "0b". For example, a nine-bit value might be shown as "0b101101111".

Examples in this document use the notation from the Unicode Standard [UNICODE3] as well as the ISO 10646 names. For example, the letter "a" may be represented as either "U+0061" or "LATIN SMALL LETTER A".

UTF-6 converts strings with internationalized characters into strings of US-ASCII that are acceptable as host name parts in current DNS host naming usage. The former are called "pre-converted" and the latter are called "post-converted". This specification defines both a forward and reverse transformation algorithm.

2. Hostname Part Transformation

According to [STD13], hostname parts must be case-insensitive, start and

end with a letter or digit, and contain only letters, digits, and the hyphen character ("-"). This, of course, excludes most characters used by non-English speakers, characters, as well as many other characters in the ASCII character repertoire. Further, domain name parts must be **63 octets or shorter in length.**

2.1 Post-Converted Name Prefix

This document defines the string 'wq--' as a prefix to identify UTF-6-encoded sequences. For the purposes of comparison in the IDN Working Group activities, the 'wq--' prefix should be used solely to identify UTF-6 sequences. However, should this document proceed beyond draft status the prefix should be changed to whatever prefix, if any, is the final consensus of the IDN working group.

Note that the prepending of a fixed identifier sequence is only one mechanism for differentiating ASCII character encoded international domain names from 'ordinary' domain names. One method, as proposed in [IDNRACE], is to include a character prefix or suffix that does not appear in any name in any zone file. A second method is to insert a domain component which pushes off any international names one or more levels deeper into the DNS heirarchy. There are trade-offs between these two methods which are independent of the Unicode to ASCII transcoding method finally chosen. We do not address the international vs. 'ordinary' name differention issue in this paper.

2.2 Hostname Prepartion

The hostname part is assumed to have at least one character disallowed by [STD13], and that is has been processed for logically equivalent character mapping, filtering of disallowed characters (if any), and compatibility composition/decomposition before presentation to the UTF-6 conversion algorithm.

While it is possible to invent a transcoding mechanism that relies on certain Unicode characters being deemed illegal within domain names and hence available to the transcoding mechanism for improving encoding efficiency, we feel that such a proposal would complicate matters excessively. We also believe that Unicode name preprocessing for both name resolution and name registration should be considered as s separate, independent issues, which we will attempt to address in a separate document.

2.3 Definitions

For clarity:

'integer' is an unsigned binary quantity;
'byte' is an 8-bit integer quantity;
'nibble' is a 4-bit integer quantity.

2.4 UTF-6 Encoding

The idea behind this scheme was to improve on the UTF-5 transformation algorithm described in [IDNDUERST] by providing a straightforward compression mechanism. UTF-6 defines a compression mechanism by indentifying identical leading byte or nibble values in the pre-converted string, and using the length of this leading value to select a mask which can be applied to the pre-converted string. The resulting post-converted string is preserves the simplicity and readability of UTF-5 while enabling longer sequences to be encoded into a single host name part.

2.4.1 Variable Length Hex Encoding

The variable length hex encoding algorithm was introduced by Duerst in [IDNDUERST]. It encodes an integer value in a slight modification of traditional hexadecimal notation, the difference being that the most significant digit is represented with an alternate set of "digits" - -- 'g through 'v' are used to represent 0 through 15. The result is a variable length encoding which can efficiently represent integers of arbitrary length.

The variable length nibble encoding of an integer, C, is defined as follows:

1. Skip over leading non-significant zero nibbles to find I, the first significant nibble of c;

2. Emit the Ith character of the set [ghijklmopqrstuv];

3. Continue from most to least significant, encoding each remaining nibble J by emitting the Jth character of the set [0123456789abcdef].

Examples:

```
0x1f4c    is encoded as "hf4c"
0x0624    is encoded as "m24"
0x0000    is encoded as "g"
'n'       a single character in single quotes stands for the
          Unicode code point for that character.
```

2.4.2 UTF-6 Compression Algorithm

UTF-6 improves on the UTF-5 encoding by providing compression, which enables encoding of a larger number of characters in each hostname part. The compression algorithm is defined as follows:

1. Set the mask to 0xFFFF;

2. If the number of non '-' characters is less than 2, proceed to step 5;

3. If the most significant byte of every non '-' character is the

same value:

3a. Set HB to this value;
3b. Emit 'Y';
3c. Emit the variable length hex encoding of HB;
3d. Set the mask to 0x00FF;
3e. Proceed to step 5.

4. If the most significant nibble of every non '-' character is the same value:

4a. Set HN to this value;
4b. Emit 'Z';
4c. Emit the variable length hex encoding of HN;
4d. Set the mask to 0x0FFF.

5. Foreach input character:

5a. Set HN to the result of the bitwise AND of the input character and the mask;
5b. Emit the variable length nibble encoding of HN.

2.4.3 Forward Transformation Algorithm

The UTF-6 transformation algorithm accepts a string in UTF-16 [ISO10646] format as input. The encoding algorithm is as follows:

1. Break the hostname string into dot-separated hostname parts. For each hostname part, perform steps 2 and 3 below;

2. Compress the component using the method described in section 2.4.2 above, and encode using the encoding described in section 2.4.1;

3. Prepend the post-converted name prefix 'wq--' (see section 2.1 above) to the resulting string.

2.5 UTF-6 Decoding

2.5.1 Variable Length Hex Decoding

1. Let N be the lower case of the first input character;

If N is not in set [ghijklmnopqrstuv] return error,
else consume the input character;

2. Let R = N - 'g';

3. If another input character exists,
then let N be the lower case of the next input character,
else goto Step 9;

4. If N is not in the set [0123456789abcdef], go to Step 9;

5. Let N = the lower case of the next input character and consume the input character;

6. Let R = R * 16;

7. If N is in set [0123456789],
   then let R = R + (N - '0'),
   else let R = R + (N - 'a') + 10;

8. Go to step 3;

9. Return decoded result R.

2.5.2 UTF-6 Decompression Algorithm

1. Let N be the lower case of the first input character;

2. If N != 'y' and N != 'z',

   2a. Let CPART be 0;
   2b. Let VMAX be 0xFFFF;

   This is the no-compression case;

3. If N == 'y',

   3a. Let M be the variable length hex decoding of the next character;
   3b. Let CPART be the result of M * 0x0100;
   3c. Let VMAX be 0x00FF;
   3d. Continue to Step 5;

4. If N == 'z',

   4a. Let M be the variable length hex decoding of the next character;
   4b. Let CPART be the result of M * 0x1000;
   4c. Let VMAX be 0x0FFF;
   4d. Continue to Step 5;

5. While another input character exists, let N be the lower case of the next input character, and do the following:

   5a. If N == '-' consume the character and
         then append '-' to the result string,
         else let VPART be the next variable hex decoded value;
   5b. If VPART > VMAX, return error,
         else append CPART + VPART to the result string;

6. Return the result string.

2.5.3 Reverse Transformation Algorithm

1. Break the string into dot-separated components and apply Steps 2 through 4 to each component:

2. Check for legality (in terms of RFC1035 permitted characters) and return error status if illegal,

3. Remove the post converted name prefix 'wq--' (see Section 2.1),

4. Decompress the component using the decompression algorithm described above.

5. Concatenate the decoded segments with dot separators and return.

3. Examples

The examples below illustrate the encoding algorithm and provide comparisons to alternate encoding schemes. UTF-5 sequences are prefixed with '----', as no ACE prefix was defined for that encoding.

3.1 'www.walid.com' (in Arabic):

UTF-16: U+0645 U+0648 U+0642 U+0639 . U+0648 U+0644 U+064A U+062F .
U+0634 U+0631 U+0643 U+0629

UTF-6: wq--ymk5k8k2j9.wq--ymk8k4kaif.wq--ymj4j1k3i9

UTF-5: ----m45m48m42m39.----m48m44m4am2f.----m34m31m43m29

RACE: bq--azcuqqrz.bq--azeeisrp.bq--ay2dcqzj

LACE: bq--aqdekscche.bq--aqdeqrckf5.bq--aqddimkdfe

3.2 Mixed Katakana and Hiragana (SOREZORENOBASHO)

UTF-16: U+305D U+308C U+305E U+308C U+306E U+5834 U+6240

UTF-6:

UTF-5:

RACE: bq--4ayf3memgbpdbdbqnzmdiysa

LACE: bq--auyf4dc7rrxacwbuafrea

3.3 Currently Disallowed ASCII Characters ($OneBillionDollars!):

UTF-16: U+0024 U+004F U+006E U+0065 U+0042 U+0069 U+006C U+006C

U+0069 U+006F U+006E U+0044 U+006F U+006C U+006C U+0061
U+0072 U+0073 U+0021

UTF-6:

UTF-5:

RACE: bq--aase74tfijuwy4djn6xei44mnrqxe5zb

LACE: bq--cmacit4omvbgs4dmnfxw5rdpnrwgc5ttee

4. Security Considerations

Much of the security of the Internet relies on the DNS and any change to the characteristics of the DNS may change the security of much of the Internet. Therefore UTF-6 makes no changes to the DNS itself.

UTF-6 is designed so that distinct Unicode sequences map to distinct domain name sequences (modulo the Unicode and DNS equivalence rules). Therefore use of UTF-6 with DNS will not negatively affect security.

5. References

[IDNCOMP] Paul Hoffman, "Comparison of Internationalized Domain Name Proposals", draft-ietf-idn-compare.

[IDNREQ] James Seng, "Requirements of Internationalized Domain Names", draft-ietf-idn-requirement.

[IDNNAMEPREP] Paul Hoffman and Marc Blanchet, "Preparation of Internationalized Host Names", draft-ietf-idn-nameprep

[IDNDUERST] M. Duerst, "Internationalization of Domain Names", draft-duerst-dns-i18n.

[ISO10646] 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. UTF-16 is described in Annex Q, published as Amendment 1. 17 other amendments are currently at various stages of standardization.

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

[STD13] Paul Mockapetris, "Domain names - implementation and specification", November 1987, STD 13 (RFC 1035).

[UNICODE3] The Unicode Consortium, "The Unicode Standard -- Version 3.0", ISBN 0-201-61633-5. Described at <http://www.unicode.org/unicode/standard/versions/Unicode3.0.html>.

A. Acknowledgements

The structure (and some of the structural text) of this document is intentionally borrowed from the LACE IDN draft (draft-ietf-idn-lace) by Mark Davis and Paul Hoffman.

The 'SOREZORENOBASHO' example was taken from draft-ietf-idn-brace draft by Adam Costello.

B. IANA Considerations

There are no IANA considerations in this document.

C. Author Contact Information

Mark Welter
Brian W. Spolarich
WALID, Inc.
State Technology Park
2245 S. State St.
Ann Arbor, MI 48104
+1-734-822-2020

mwelter@walid.com
briansp@walid.com
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.0.1 (GNU/Linux)
Comment: For info see http://www.gnupg.org

iD8DBQE6FaCt/DkPcNgtD/0RAtRmAJwISVeJGY6qmll71mL+Axc51o8iIwCgmNt/
86RcQh1JQYWTux+8FS+XvMU=
=bxiv
-----END PGP SIGNATURE-----