Internet Draft
draft-hoffman-idn-cidnuc-03.txt
March 10, 2000
Expires in six months

Paul Hoffman
IMC & VPNC

Compatible Internationalized Domain Names Using Compression

Abstract

This protocol describes a transformation method for representing non-ASCII characters in domain names in a fashion that is completely compatible with the current DNS. It meets the many requirements for internationalization of domain names.

Note: this protocol is quite experimental and should not be deployed in the Internet until it reaches standards track in the IETF.

1. Introduction

There is a strong world-wide desire to use characters other than plain ASCII in domain names. Domain names have become the equivalent of business or product names for many services on the Internet, so there is a need to make them usable by people whose native scripts are not representable by ASCII. The requirements for internationalizing domain names are described in [IDNReq].

The protocol in this document describes how to take almost any character used in human writing and use it in a domain name in a way that is completely compatible with the current DNS. The protocol requires absolutely no changes to the DNS [STD13].

The protocol works for both entry and display of internationalized characters. For domain name entry, a user enters the international (that is, non-ASCII) characters of a domain name into a converter, and that converter transforms the name entered into a DNS-compatible format. Each domain part of internationalized domain names is tagged, and some parts may be internationalized while others use today's plain ASCII format. For domain name display, the display utility converts each tagged domain part from its DNS-compatible format into the internationalized characters and displays them inline with any non-internationalized domain part. Users never have to see the converted versions of the internationalized name parts.

In formal terms, this protocol describes a character encoding scheme of the ISO 10646 [ISO10646] coded character set and the rules for using that scheme in the DNS. As such, it could also be called a "charset" as defined in [RFC2278].

The protocol has the following features:

- There are no changes to the DNS protocols or the way that domain names are interpreted. There are also no change to the DNS root servers, nor to zone files. The protocol can start to be used the DNS today with only changes in the non-protocol portions.

- There is exactly one way to convert internationalized domain parts to and from DNS-compatible parts. Domain part uniqueness is preserved. Domain parts that have no international characters are not changed.

- Essentially all characters written in all common (and many uncommon) human scripts can be put in any name part.

- Transformed names can be entered and displayed without ASCII case sensitivity.

- Names using this protocol can include more internationalized characters than with other ASCII-converted protocols that have been suggested to date.

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".

This protocol converts strings with internationalized characters into strings of US-ASCII that are acceptable as domain name parts in current DNS host naming usage. The former are called "pre-converted" and the latter are called "post-converted".

2. Domain Part Transformation

Any domain part that contains one or more non-ASCII characters is transformed into a DNS-compatible name before passing it to a DNS resolver or other program that uses traditional domain names. This step is usually done at the time a user enters a domain name into an application. When a domain name is displayed to a user, the display program can covert any domain part that is tagged as holding internationalized characters into a displayable representation that includes the internationalized characters.

It is important to note that the following sections contain many normative statements with "MUST" and "MUST NOT". Any implementation that does not follow these statements exactly is likely to cause damage to the Internet by creating non-unique representations of domain names.

According to [STD13], domain parts must be case-insensitive, start with a letter, and contain only letters, digits, and the hyphen character ("-"). This, of course, excludes any internationalized 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 Name tagging

Internationalized domain parts are converted to and from a display representation that include non-ASCII characters. Thus, a program that converts from DNS-compatible name parts to viewable name parts must be able to recognize name parts that need to be converted.

All post-converted name parts that contain internationalized characters begin with the string "aq8". (Of course, because domain name parts are case-insensitive, this might also be represented as "Aq8" or "aQ8" or "AQ8".) The string "aq8" was chosen because it is extremely unlikely to exist in domain parts before this specification was produced. As a historical note, in early March 2000, none of the second-level domain parts in any of the .com, .edu, .net, and .org top-level domains began with "aq8"; there are about 9,500 other strings of three legal characters that have this property and could be used instead.

Note that a zone administrator can still choose to use "aq8" at the beginning of a domain part even if that part does not contain internationalized characters. Zone administrators SHOULD NOT create domain part names that begin with "aq8" unless those names are post-converted names. Creating domain part names that begin with "aq8" but that are not post-converted names may cause display systems that conform to this document to display the name parts in a possibly-confusing fashion to users. However, creating such names will not cause any DNS resolution problems; it will only cause display problems (and possibly entry problems) for some users.

2.2 Converting an internationalized name to a domain name part

To convert a string of internationalized characters into a DNS-compatible domain name part, the following steps MUST be preformed in the exact order of the subsections given here. Note that these steps MUST be done by zone administrators who are creating internationalized domain name parts in their zones and MUST be done by clients who are resolving domain names.

The input name string consists of characters from the ISO 10646 character set in big-endian UTF-16 encoding. This is the pre-converted string.

Characters outside the first plane of characters (that is, outside the first 0xFFFF characters) MUST be represented using surrogates, as described in the UTF-16 description in ISO 10646.

The characters in Table 1 MUST NOT appear in pre-converted domain name parts. The characters in this list have been chosen for many reasons, mostly to avoid problems with displayed characters. The reasons include:

- The character is a period

- The character is a separator (space, line, or paragraph)

- The character is a control character

- The character is a formatting character

- The character is a private-use character

Table 1: Characters illegal in domain names
U+002E (FULL STOP)
All characters in the Unicode Character Database [UnicodeData] whose General Category is any of:
Zs
Zl
Zp
Cc
Cf
Co

Design note: The above list will proabably change and will probably be taken to a separate document so there can be more focused discussion on it. For example, there appears to be a desire to not allow uppercase and lowercase, and some discussion of not allowing characters that do not "normally" appear in "names". The above list could include all of the characters of the not- chosen case and of type "punctuation" and "symbol", minus those that "normally" appear in "names".

Design note: There is no reason to assume that this database must be run by the Unicode Consortium. It is quite believable that, given the importance of the database, that it could be maintained by IANA for the IETF, quite probably with the help of the Unicode Consortium.

2.2.1 Check for a name that cannot be transformed

An untransformed input strings that is already a legitimate domain name part MUST NOT be converted. Each character in the input string MUST be compared to the following list of characters:

```
U+002D (HYPHEN-MINUS)
U+0030 through U+0039 (DIGIT ZERO, ...)
U+0041 through U+005A (LATIN CAPITAL LETTER A, ...)
U+0061 through U+007A (LATIN SMALL LETTER A, ...)
```

If all the characters in the input string are in the above set of characters, the conversion MUST stop with an error. The input string itself MUST be used as the domain name part.

2.2.2 Check for illegal characters in the input string

Each character in the input string MUST be checked against Table 1. If any character in the input string matches a character listed in Table 1, the conversion MUST stop with an error. The characters in Table 1 MUST NOT appear in any internationalized domain name part.

Further, each character in the input string MUST be checked to see if it is part of a malformed surrogate pair. If any character is part of a malformed surrogate pair, the conversion MUST stop with an error. Malformed surrogate pairs MUST NOT appear in any internationalized domain name part.

2.2.3 Normalize the input string

The entire input string MUST be normalized using Normalization Form C as described in [Norm]. The normalization MUST be applied to the entire input string, not to substrings. The result of this step is the normalized string.

2.2.4 Compress the normalized string

The entire normalized string MUST be compressed using the compression algorithm specified in section 2.4. The result of this step is the compressed string.

2.2.5 Check the length of the compressed string

The compressed string MUST be 37 octets or shorter. If the compressed string is 38 octets or longer, the conversion MUST stop with an error.

2.2.6 Encode the compressed string with Base32

The compressed string MUST be converted to a DNS-compatible encoding using the Base32 encoding described in section 2.5. The result of this step is the encoded string.

2.2.7 Prepend "aq8" to the encoded string and finish

Prepend the characters "aq8" to the encoded string. This is the domain name part that can be used in DNS resolution.

2.3 Converting a domain name part to an internationalized name

The input string for conversion is a valid domain name part.

2.3.1 Strip the "aq8"

The input string MUST begin with the characters "aq8". If it does not, the conversion MUST stop and the displaying program MUST NOT treat the domain name part as internationalized characters and the input string is the post-converted string. Otherwise, remove the characters "aq8" from the input string. The result of this step is the stripped string.

2.3.2 Decode the stripped string with Base32

The entire stripped string MUST be checked to see if it is valid Base32 output. The entire stripped string MUST changed to all lower-case letters. If any resulting characters are not in Table 2, the conversion MUST stop and the displaying program MUST NOT treat the domain name part as internationalized characters; the input string is the post-converted string. Otherwise, the entire resulting string MUST be converted to a binary format using the Base32 decoding described in section 2.5. The result of this step is the decoded string.

2.3.3 Decompress the decoded string

The entire decoded string MUST be converted to ISO 10646 characters using the decompression algorithm described in section 2.4. The result of this is the internationalized string.

2.3.4 Verify the internationalized string and finish

Each character in the internationalized string MUST be verified before the string can be used. If the string only consists of the characters listed in section 2.2.1, the conversion MUST stop and the input string is the post-converted string. If any of the characters in the string are Table 1 from section 2.2.2, the conversion MUST stop and the input string is the post-converted string.

The internationalized string MUST be checked for invalid surrogate pairs, as described in ISO 10646. If an invalid surrogate pair is found, the conversion MUST stop and the input string is the post-converted string.

If no errors are found, the verified string is the post-converted string.

2.4 Compression algorithm

The basic method for compression is to reduce sequences of characters that all have the same upper octet to single octets. Any string that has a character that doesn't have the same upper octet as all the other characters in the string has all the octets of the input string in the output string.

The compressed string always has a one-octet header. For one-octet mode, the header octet is the upper octet of the stream. For two-octet mode, the header octet is 0xD8, which is the upper octet of a surrogate pair. Design note: It is impossible to have a legal stream of UTF-16 characters that has all the upper octets being 0xD8 because a character whose upper octet is 0xD8 must be followed by one whose upper octet is in the range 0xDC through 0xDF.

Although the two-octet mode limits the number of characters to 17, this is still generally enough for almost all names in almost scripts. Also, this limit is close to the limits set by other encoding proposals.

Note that all name parts whose characters have the same upper octet MUST be expressed in the one-octet mode. This requirement prevents a single domain name part from having two encodings.

2.4.1 Compressing a string

Design note: No checking is done on the input to this algorithm. It is assumed that all checking for valid ISO 10646 characters has already been done by a previous step in the conversion process.

1) Read each character in the input stream, comparing the upper octet of each. If all of the upper octets match, go to step 3.

2) Output 0xD8, followed by the entire input stream. Finish.

3) Output the upper octet of the first character. Output the lower octet of each character in the input. Finish.

2.4.2 Decompressing a string

1) Read the first octet of the input string. If it is 0xD8, go to step

3.

2) Call the value of this first octet "upper". For each other octet in the input, output "upper", then output the octet from the input. Finish.

3) Read the rest of the input stream and put it in the output stream. Finish.

2.5 Base32

In order to encode non-ASCII characters in DNS-compatible domain parts, they must be converted into legal characters. This is done with Base32 encoding, described here.

Table 2 shows the mapping between input bits and output characters in Base32. Design note: the digits used in Base32 are "2" through "7" instead of "0" through "6" in order to avoid digits "0" and "1". This helps reduce errors for users who are entering a Base32 stream and may misinterpret a "0" for an "O" or a "1" for an "l".

Table 2: Base32 conversion

bits	char	hex	bits	char	hex
00000	a	0x61	10000	q	0x71
00001	b	0x62	10001	r	0x72
00010	c	0x63	10010	s	0x73
00011	d	0x64	10011	t	0x74
00100	e	0x65	10100	u	0x75
00101	f	0x66	10101	v	0x76
00110	g	0x67	10110	w	0x77
00111	h	0x68	10111	x	0x78
01000	i	0x69	11000	y	0x79
01001	j	0x6a	11001	z	0x7a
01010	k	0x6b	11010	2	0x32
01011	l	0x6c	11011	3	0x33
01100	m	0x6d	11100	4	0x34
01101	n	0x6e	11101	5	0x35
01110	o	0x6f	11110	6	0x36
01111	p	0x70	11111	7	0x37

2.5.1 Encoding octets as Base32

The input is a stream of octets. However, the octets are then treated as a stream of bits.

Design note: The assumption that the input is a stream of octets (instead of a stream of bits) was made so that no padding was needed. If you are reusing this encoding for a stream of bits, you must add a padding mechanism in order to differentiate different lengths of input.

1) Set the read pointer to the beginning of the input bit stream.

2) Look at the five bits after the read pointer. If there are not five bits, go to step 5.

3) Look up the value of the set of five bits in the bits column of Table 2, and output the character from the char column (whose hex value is in the hex column).

4) Move the read pointer five bits forward. If the read pointer is at the end of the input bit stream (that is, there are no more bits in the input), stop. Otherwise, go to step 2.

5) Pad the bits seen until there are five bits.

6) Look up the value of the set of five bits in the bits column of Table 2, and output the character from the char column (whose hex value is in the hex column).

2.5.2 Decoding Base32 as octets

The input is octets in network byte order. The input octets MUST be values from the second column in Table 2.

1) Set the read pointer to the beginning of the input octet stream.

2) Look up the character value of the octet in the char column (or hex value in hex column) of Table 2, and output the five bits from the bits column.

3) Move the read pointer one octet forward. If the read pointer is at the end of the input octet stream (that is, there are no more octets in the input), stop. Otherwise, go to step 2.

2.5.3 Base32 example

Assume you want to the value 0x3a270f93. The bit string is:

```
3   a    2   7    0   f    9   3
00111010 00100111 00001111 10010011
```

Broken into chunks of five bits, this is:

```
00111 01000 10011 10000 11111 00100 11
```

The output of encoding is:

```
00111 01000 10011 10000 11111 00100 11
  h     i     t     q     7     e    y
or "hitq7ey".
```

3. Implementing User Interfaces

This section gives guidelines to creators of programs that allow entry or display of domain names.

The use of internationalized domain name parts in user applications should be as transparent to the user as possible. A user should be able to enter and see internationalized domain names as the pre-converted names if at all possible.

For instance, if the user is able to enter Chinese characters anywhere in a program, he or she should also be able to enter Chinese characters into any interface component that would take in a domain name, such as dialog box asking for a URL. Similarly, if any part of a program can display Arabic characters, any domain name that has Arabic characters in it should be able to be displayed with Arabic characters, not as the ASCII transformation of those characters.

3.1 Name entry

In non-internationalized systems, the user enters a domain name and that name is usually sent unchecked to a domain name resolver, which returns an IPv4 address. With internationalized names, the user application MUST convert the pre-converted name into a post-converted name so that is acceptable to resolvers.

Some users might have access to the post-converted format of an internationalized name. Because of this, users SHOULD be able to enter post-converted names directly into an interface component for domain names. This capability should already be in the interface because the post-converted names are already legal. It is important that interfaces not prohibit the entry of long domain names. (Of course, they should not be prohibiting them anyway.)

There are a wide variety of user input methods. Keyboard input methods vary widely from script to script, and even within a single script, there are often more than one method. Humorously, people who don't use a particular script often cannot comprehend how someone who uses that script can input it with a keyboard and will often declare such input as impossible.

Regardless of the input method, any system that allows input of non-ASCII characters SHOULD allow input of pre-converted domain names in the same fashion.

3.2 Name display

As a user enters internationalized characters, they are often displayed to the user at the same time. For instance, in a typical entry box for a URL, the characters are displayed as they are entered. Such display should, of course, also happen for internationalized characters in domain names.

Choosing what to do with domain names in free text is more difficult because not all scripts are easily displayable. For instance, assume that you are reading a sentence on a page that says "You can reach the company at" followed by a URL. If the domain name portion of the URL is internationalized, each domain name part SHOULD be shown as a pre-converted string if possible. If it is not possible (such as if no font for the script is available), the domain name part SHOULD be shown as post-converted characters.

A display program has two choices when displaying an internationalized name part for which there is one or more characters that the program cannot display. The first choice is to display (but not replace with) a "replacement character" that does not look like any other character in the display. The second choice is to display the post-converted name; this is admittedly ugly and does not give the user any useful information other than "the text could not be displayed". In general, the first option is the better choice. However, it is very important that user still be able to copy a domain name part even if it has characters that cannot be displayed. Thus, if a display system chooses to display a "replacement character", the underlying character MUST still be the undisplayable character.

Note that some domain name parts that start with "aq8" are not pre-converted parts. Such names may contain characters that are not in the Base32 character set. In such cases, a display program SHOULD display the name part without attempting to convert it to post-converted characters.

3.3 Zone files and registration

Historically, zone files have been maintained as US-ASCII text. For stability reasons, this practice MUST continue. Thus, zone files MUST only contain post-converted name parts. Zone administrators can use tools to enter and view the internationalized parts of zone files.

Registrars for public name spaces such as .com have the same requirements as any zone administrator. They MUST be sure not to register names that are illegal. In the case if this protocol, the registration can continue to be done with US-ASCII characters, but the registrar MUST then check that the conversion to an internationalized name does not result in an error.

3.4 Users and post-converted name parts

The Internet has a long history of trying to hide technical detail from users only to have that detail exposed, often in a confusing fashion. This protocol attempts to minimize the impact of such exposure.

Clearly, no user will be able to understand a post-converted name part. However, they are unlikely to have any significant problems with them. It is likely to become common lore that domain names that have internationalized parts also have an all-text version that looks like gibberish. These long names can be copied (by program or even by hand) just like current domain names are.

4. Security Considerations

Much of the security of the Internet relies on the DNS. Thus, any change to the characteristics of the DNS can change the security of much of the Internet. Thus, this protocol makes no changes to the DNS itself.

Host names are used by users to connect to Internet servers. The security of the Internet would be compromised if a user entering a single internationalized name could be connected to different servers based on different interpretations of the internationalized domain name.

This protocol is designed so that every internationalized domain part can be represented as one and only one DNS-compatible string. If there is any way to follow the steps in this document and get two or more different results, it is a severe and fatal error in the protocol.

5. References

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

[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. [[[THIS REFERENCE NEEDS TO BE UPDATED AFTER DETERMINING ACCEPTABLE WORDING]]]

[Norm] Mark Davis and Martin Duerst, "UCharacter Normalization in ITEF Protocols",
draft-duerst-i18n-norm.

[RFC2045] Ned Freed and Nathaniel Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", November 1996, RFC 2045.

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

[RFC2278] Ned Freed and Jon Postel, "IANA Charset Registration Procedures", January 1998, RFC 2278.

[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>.

[UnicodeData] The Unicode Character Database, <ftp://ftp.unicode.org/Public/UNIDATA/UnicodeData.txt>. The database is described in <ftp://ftp.unicode.org/Public/UNIDATA/UnicodeCharacterDatabase.html>.

A. Acknowledgements

Mark Davis contributed many ideas to the initial draft of this document. Graham Klyne and Martin Duerst offered technical comments on the algorithms used.

Base32 is quite obviously inspired by the tried-and-true Base64 Content-Transfer-Encoding described in [RFC2045].

B. Changes from Previous Versions of this Draft

B.1 Changes from -02 to -03

Throughout: changed "wg4" to "aq8".

2.2: Updated the first design note to indicate that the table will probably be moved to its own draft.

2.2.3: Changed reference for normalization from [UTR15] to [Norm].

5: Updated the reference for [IDNReq]. Removed [UTR15] and replaced it with [Norm].

B.2 Changes from -01 to -02

Throughout: Changed "ph6" to "wg4".

2.1: Updated count of unused three-letter prefixes.

2.3: Removed all the error states and clarified that any error in conversion means that the input string is the post-converted string.

2.4: Radically changed the compression scheme; the previous one was far too cumbersome.

2.5: Renumbered Table 3 to Table 2.

2.5.1: Changed the second paragraph (should have been done in the change to -01 to remove padding).

3.2: Clarified the paragraph emphasizing the need for users to be able to copy names even if they are not displayable.

5: Removed reference to [UTR6].

A: Added Martin Duerst. Removed reference to the compression algorithm because it has changed.

B.3 Changes from -00 to -01

Throughout: Changed references to the character set from Unicode to ISO 10646, even though they are equivalent. Also changed references to the rules for surrogate pairs to ISO 10646.

1.1: Clarified last paragraph.

2.2: Reworded the first design note to make excluding case stuff more likely.

2.5: Removed the "8" padding in the Base32 algorithm because it was superfluous.

2.5.1: Removed "in network byte order" from the first sentence because it was redundant.

3.3: Made the first paragraph stronger.

5: Added reference to ISO 10646. This still needs work.

A: Added Graham Klyne.

C. IANA Considerations

There are no IANA considerations in the current draft. However, if it is decided to have IANA maintain the character database, this section will become much longer.

D. Author Contact Information

Paul Hoffman
Internet Mail Consortium and VPN Consortium
127 Segre Place
Santa Cruz, CA 95060 USA
paul.hoffman@imc.org and paul.hoffman@vpnc.org