IDNABIS
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2010

P. Resnick, Ed.
Qualcomm Incorporated
P. Hoffman
VPN Consortium

July 3, 2009

Mapping Characters in IDNA
draft-ietf-idnabis-mappings-01

Status of this Memo

Copyright Notice

Abstract

In the original version of the Internationalized Domain Names in Applications (IDNA) protocol, any Unicode code points taken from user input were mapped into a set of Unicode code points that "make sense", which were then encoded and passed to the domain name system (DNS). The current version of IDNA presumes that the input to the protocol comes from a set of "permitted" code points, which it then encodes and passes to the DNS, but does not specify what to do with the result of user input. This document describes the actions taken by an implementation between user input and passing permitted code points to the new IDNA protocol.

1. Introduction

This document describes the operations that can be applied to user input in order to get it into a form acceptable by the Internationalized Domain Names in Applications (IDNA) protocol [I-D.ietf-idnabis-protocol]. The document describes the underlying architectural principles (in section 2 and the general implementation procedure (in section 3).

It should be noted that this document does not specify the behavior of a protocol that appears "on the wire". It describes an operation that is to be applied to user input in order to prepare that user input for use in an "on the network" protocol. As unusual as this may be for an IETF protocol document, it is a necessary operation to maintain interoperability.

2. Architectural Principles

An application that implements the IDNA protocol [I-D.ietf-idnabis-protocol] will always take any user input and convert it to a set of Unicode code points. That user input may be acquired by any of several different input methods, all with

differing conversion processes to be taken into consideration (e.g., typed on a keyboard, written by hand onto some sort of digitizer, spoken into a microphone and interpreted by a speech-to-text engine, etc.). The process of taking any particular user input and mapping it into a Unicode code point may be a simple one: If a user strikes

the "A" key on a US English keyboard, without any modifiers such as the "Shift" key held down, in order to draw a Latin small letter A ("a"), many (perhaps most) modern operating system input methods will produce to the calling application the code point U+0061, encoded in a single octet.

Sometimes the process is somewhat more complicated: a user might strike a particular set of keys to represent a combining macron followed by striking the "A" key in order to draw a Latin small letter A with a macron above it. Depending on the operating system, the input method chosen by the user, and even the parameters with which the application communicates with the input method, the result might be the code point U+0101 (encoded as two octets in UTF-8 or UTF-16, four octets in UTF-32, etc.), the code point U+0061 followed by the code point U+0304 (again, encoded in three or more octets, depending upon the encoding used) or even the code point U+FF41 followed by the code point U+0304 (and encoded in some form). And these examples leave aside the issue of operating systems and input methods that do not use Unicode code points for their character set.

In every case, applications (with the help of the operating systems on which they run and the input methods used) need to perform a mapping from user input into Unicode code points.

The original version of the IDNA protocol [RFC3490] used a model whereby input was taken from the user, mapped (via whatever input method mechanisms were used) to a set of Unicode code points, and then further mapped to a set of Unicode code points using the Nameprep profile specified in [RFC3491]. In this procedure, there are two separate mapping steps: First, a mapping done by the input method (which might be controlled by the operating system, the application, or some combination) and then a second mapping performed by the Nameprep portion of the IDNA protocol. The mapping done in Nameprep includes a particular mapping table to re-map some characters to other characters, a particular normalization, and a set of prohibited characters.

Note that the result of the two step mapping process means that the mapping chosen by the operating system or application in the first step might differ significantly from the mapping supplied by the Nameprep profile in the second step. This has advantages and disadvantages. Of course, the second mapping regularizes what gets looked up in the DNS, making for better interoperability between implementations which use the Nameprep mapping. However, the application or operating system may choose mappings in their input methods, which when passed through the second (Nameprep) mapping result in characters that are "surprising" to the end user.

The other important feature of the original version of the IDNA protocol is that, with very few exceptions, it assumes that any set of Unicode code points provided to the Nameprep mapping can be mapped into a string of Unicode code points that are "sensible", even if that means mapping some code points to nothing (that is, removing the code points from the string). This allowed maximum flexibility in input strings.

The present version of IDNA differs significantly in approach from the original version. First and foremost, it does not provide explicit mapping instructions. Instead, it assumes that the application (perhaps via an operating system input method) will do whatever mapping it requires to convert input into Unicode code points. This has the advantage of giving flexibility to the application to choose a mapping that is suitable for its user given specific user requirements, and avoids the two-step mapping of the original protocol. Instead of a mapping, the current version of IDNA provides a set of categories that can be used to specify the valid code points allowed in a domain name.

In principle, an application ought to take user input of a domain name and convert it to the set of Unicode code points that represent the domain name the user intends. As a practical matter, of course, determining user intent is a tricky business, so an application needs to choose a reasonable mapping from user input. That may differ based on the particular circumstances of a user, depending on locale, language, type of input method, etc. It is up to the application to make a reasonable choice.

3. The General Procedure

This section defines a general algorithm that applications ought to implement in order to produce Unicode code points that will be valid under the IDNA protocol. An application might implement the full mapping as described below, or can choose a different mapping. In fact, an appliction might want to implement a full mapping that is substantially compatible with the original IDNA protocol instead of the algorithm given here.

The general algorithm that an application (or the input method provided by an operating system) ought to use is relatively straightforward and generally follows section 5 of [I-D.ietf-idnabis-protocol]:

1. All characters are mapped using Unicode Normalization Form C (NFC).

2. Upper case characters are mapped to their lower case equivalents by using the algorithm for mapping Unicode characters.

3. Full-width and half-width characters (those defined with Decomposition Types <wide> and <narrow>) are mapped to their decomposition mappings as shown in the Unicode character database.

Definitions for the rules in this algorithm can be found in [Unicode51]. Specifically:

- Unicode Normalization Form C can be found in Annex #15 of [Unicode51].

- In order to map upper case characters to their lower case equivalents (defined in section 3.13 of [Unicode51]), first map characters to the "Lowercase_Mapping" property (the "<lower>" entry in the second column) in <http://www.unicode.org/Public/UNIDATA/SpecialCasing.txt>, if any. Then, map characters to the "Simple_Lowercase_Mapping" property (the fourteenth column) in <http://www.unicode.org/Public/UNIDATA/UnicodeData.txt>, if any.

- In order to map full-width and half-width characters to their decomposition mappings, map any character whose "Decomposition_Type" (contained in the first part of of the sixth column) in <http://www.unicode.org/Public/UNIDATA/UnicodeData.txt> is either "<wide>" or "<narrow>" to the "Decomposition_Mapping" of that character (contained in the second part of the sixth column) in <http://www.unicode.org/Public/UNIDATA/UnicodeData.txt>.

- The <http://www.unicode.org/Public/UNIDATA/UCD.html> web page has useful descriptions of the contents of these files.

If this mappings in this document are applied to versions of Unicode later than Unicode 5.1, the later versions of the Unicode Standard should be consulted.

These are a minimal set of mappings that an application should strongly consider doing. Of course, there are many others that might be done.

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

This document suggests creating mappings that might cause confusion for some users while alleviating confusion in other users. Such confusion is not covered in any depth in this document (nor in the other IDNA-related documents).

6. Normative References

[I-D.ietf-idnabis-protocol]
Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", draft-ietf-idnabis-protocol-12 (work in progress), May 2009.

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[RFC3491] Hoffman, P. and M. Blanchet, "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)", RFC 3491, March 2003.

[Unicode51] The Unicode Consortium, "The Unicode Standard, Version 5.1.0", 2008.

defined by: The Unicode Standard, Version 5.0, Boston, MA, Addison-Wesley, 2007, ISBN 0-321-48091-0, as amended by Unicode 5.1.0 (<http://www.unicode.org/versions/Unicode5.1.0/>).

Authors' Addresses

Peter W. Resnick (editor)
Qualcomm Incorporated
5775 Morehouse Drive
San Diego, CA 92121-1714
US

Phone: +1 858 651 4478
Email: presnick@qualcomm.com
URI: http://www.qualcomm.com/~presnick/

Paul Hoffman
VPN Consortium
127 Segre Place
Santa Cruz, CA 95060
US

Phone: 1-831-426-9827
Email: paul.hoffman@vpnc.org