Internationalizing Host Names in Applications (IDNA)

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.

Abstract

Until now, there has been no standard way for host names to use characters outside the ASCII repertoire. This document describes a mechanism called IDNA that enables internationalized host names, that is, host names that use characters drawn from a much larger repertoire. (The "D" in the name originally stood for "domain", but the work is actually focused on host names, so the word "host" is used throughout this document.)

1. Introduction

IDNA works by allowing applications to use certain ASCII name labels (beginning with a special prefix) to represent non-ASCII name labels. Lower-layer protocols need not be aware of this; therefore IDNA does not require changes to any infrastructure. In particular, IDNA does not require any changes to DNS servers, resolvers, or protocol elements, because the ASCII name service provided by the existing DNS is entirely sufficient.

This document does not require any applications to conform to IDNA, but applications can elect to use IDNA in order to support IDN while maintaining interoperability with existing infrastructure. Adding IDNA support to an existing application entails changes to the application only, and leaves room for flexibility in the user interface.

A great deal of the discussion of IDN solutions has focused on transition issues and how IDN will work in a world where not all of the components have been updated. Other proposals would require that user applications, resolvers, and DNS servers be updated in order for a user...
to use an internationalized host name. Rather than require widespread
updating of all components, IDNA requires only user applications to be
updated; no changes are needed to the DNS protocol or any DNS servers or
the resolvers on user’s computers.

This document is being discussed on the ietf-idna@mail.apps.ietf.org
mailing list. To subscribe, send a message to
ietf-idna-request@mail.apps.ietf.org with the single word "subscribe" in
the body of the message.

2 Terminology

[[ Editor’s note: the author’s are considering changing "host name" to
"domain name" throughout the document after discussing this further
with the DNS experts. ]]

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

A code point is an integral value associated with a character in a coded
character set.

Unicode [UNICODE] is a coded character set containing tens of thousands
of characters. A single Unicode code point is denoted by "U+" followed
by four to six hexadecimal digits, while a range of Unicode code points
is denoted by two hexadecimal numbers separated by ".", with no
prefixes.

ASCII means US-ASCII, a coded character set containing 128 characters
associated with code points in the range 0..7F. Unicode is an extension
of ASCII: it includes all the ASCII characters and associates them with
the same code points.

The term "LDH code points" is defined in this document to mean the code
points associated with ASCII letters, digits, and the hyphen-minus; that
is, U+002D, 30..39, 41..5A, and 61..7A. "LDH" is an abbreviation for
"letters, digits, hyphen".

A host label is an individual part of a host name. Host labels are
usually shown separated by dots; for example, the host name
"www.example.com" is composed of three host labels: "www", "example",
and "com". In IDNA, not all text strings can be host labels. A string
can be a host label if and only if the ToASCII operation (see section 4)
does not fail when applied to it. (The zero-length root label that is
implied in host names, as described in [STD13], is not considered a
label in this specification.)

An "ACE label" is defined in this document to be a host label that
contains only ASCII characters but represents a label containing
non-ASCII characters (ACE stands for "ASCII-compatible encoding").
Internationalized host labels generally contain non-ASCII characters,
but for every host label that cannot be directly represented in ASCII
there is an equivalent ACE label. The conversion of host labels to and
from the ACE form is specified in section 4.

The "ACE prefix" is defined in this document to be a string of ASCII
characters that appears at the beginning of every ACE label. It is
specified in section 5.
A "host name slot" is defined in this document to be a protocol element or a function argument or a return value (and so on) explicitly designated for carrying a host name. Examples of host name slots include: the QNAME field of a DNS query; the name argument of the gethostbyname() library function; the part of an email address following the at-sign (@) in the From: field of an email message header; and the host portion of the URI in the src attribute of an HTML <IMG> tag. General text that just happens to contain a host name is not a host name slot; for example, a host name appearing in the plain text body of an email message is not occupying a host name slot.

An "internationalized host name slot" is defined in this document to be a host name slot explicitly designated for carrying an internationalized host name as described in this document. The designation may be static (for example, in the specification of the protocol or interface) or dynamic (for example, as a result of negotiation in an interactive session).

A "generic host name slot" is defined in this document to be any host name slot that is not an internationalized host name slot. Obviously, this includes any host name slot whose specification predates IDNA.

3. Requirements

IDNA conformance means adherence of the following three rules:

1) Whenever a host name is put into a generic host name slot, every label MUST contain only ASCII characters. Given any host name, an equivalent host name satisfying this requirement can be obtained by applying the ToASCII operation (see section 4) to each label.

2) ACE labels SHOULD be hidden from users whenever possible. Therefore, before a host name is displayed to a user or is output into a context likely to be viewed by users, the ToUnicode operation (see section 4) SHOULD be applied to each label. When requirements 1 and 2 both apply, requirement 1 takes precedence.

3) Whenever two host labels are compared, they MUST be considered to match if and only if their ASCII forms (obtained by applying ToASCII) match using a case-insensitive ASCII comparison.

4. Conversion operations

This section specifies the ToASCII and ToUnicode operations. Each one operates on a sequence of Unicode code points (but remember that all ASCII code points are also Unicode code points). When host names are represented using character sets other than Unicode and ASCII, they will need to first be transcoded to Unicode before these operations can be applied, and might need to be transcoded back afterwards.

4.1 ToASCII

The ToASCII operation takes a sequence of Unicode code points and transforms it into a sequence of code points in the ASCII range (0..7F). The original sequence and the resulting sequence are equivalent host labels.

ToASCII fails if any step of it fails. Failure means that the original
sequence cannot be used as a host label.

ToASCII never alters a sequence of code points that are all in the ASCII range to begin with (although it may fail).

ToASCII consists of the following steps:

1. If all code points in the sequence are in the ASCII range (0..7F) then skip to step 3.

2. Perform the steps specified in [NAMEPREP].

3. Host-specific restrictions:
   Host names have additional restrictions:
   * Verify the absence of non-LDH ASCII code points; that is, the absence of 0..2C, 2E..2F, 3A..40, 5B..60, and 7B..7F.
   * Verify the absence of leading and trailing hyphen-minus; that is, the absence of U+002D at the beginning and end of the sequence.

4. If all code points in the sequence are in the ASCII range (0..7F), then skip to step 8.

5. Verify that the sequence does NOT begin with the ACE prefix.

6. Encode the sequence using the encoding algorithm in [AMC-ACE-Z].

7. Prepend the ACE prefix.

8. Verify that the number of code points is in the range 1 to 63 inclusive.

4.2 ToUnicode

The ToUnicode operation takes a sequence of Unicode code points and returns a sequence of Unicode code points. If the input sequence is a host label in ACE form, then the result is an equivalent host label that is not in ACE form, otherwise the original sequence is returned unaltered.

ToUnicode never fails. If any step fails, then the original input sequence is returned immediately in that step.

1. If all code points in the sequence are in the ASCII range (0..7F) then skip to step 3.

2. Perform the steps specified in [NAMEPREP]. (If step 3 of ToASCII is also performed here, it will not affect the overall behavior of ToUnicode, but it is not necessary.)

3. Verify that the sequence begins with the ACE prefix, and save a copy of the sequence.

4. Remove the ACE prefix.

5. Decode the sequence using decoding algorithm in [AMC-ACE-Z]. Save a copy of the result of this step.

6. Apply ToASCII.
7. Verify that the sequence matches the saved copy from step 3, using a case-insensitive ASCII comparison.

8. Return the saved copy from step 5.

5. ACE prefix

The ACE prefix, used in the conversion operations (section 4), will be specified in a future revision of this document. It will be two alphanumeric ASCII characters followed by two hyphen-minuses. The ToASCII and ToUnicode operations MUST recognize the ACE prefix in a case-insensitive manner.

For example, the eventual ACE prefix might be the string "jk--". In this case, an ACE label might be "jk--r3c2a-qc902xs", where "r3c2a-qc902xs" is the part of the ACE label that is generated by the encoding steps in [AMC-ACE-Z].

6. Implications for typical applications using DNS

In IDNA, applications perform the processing needed to input internationalized host names from users, display internationalized host names to users, and process the inputs and outputs from DNS and other protocols that carry host names.

The components and interfaces between them can be represented pictorially as:

```
+------+
| User |
+------+

^       ^
| Input and display: local interface methods |
| (pen, keyboard, glowing phosphorus, ...) |

v

Application
(conversion between local character set and Unicode is done here)

^       ^
| Call to resolver: |
| ACE |
| v |
| Resolver |
| +-------+ |
| v |

^       ^
| DNS protocol: |
| ACE |
| v |

+--------+
| DNS servers |
+--------+

+--------+
| Application servers |
+--------+
```
6.1 Entry and display in applications

Applications can accept host names using any character set or sets desired by the application developer, and can display host names in any charset. That is, the IDNA protocol does not affect the interface between users and applications.

An IDNA-aware application can accept and display internationalized host names in two formats: the internationalized character set(s) supported by the application, and as an ACE label. Applications MAY allow input and display of ACE labels, but are not encouraged to do so except as an interface for special purposes, possibly for debugging. ACE encoding is opaque and ugly, and should thus only be exposed to users who absolutely need it. The optional use, especially during a transition period, of ACE encodings in the user interface is described in section 6.4. Because name labels encoded as ACE name labels can be rendered either as the encoded ASCII characters or the proper decoded characters, the application MAY have an option for the user to select the preferred method of display; if it does, rendering the ACE SHOULD NOT be the default.

Host names are often stored and transported in many places. For example, they are part of documents such as mail messages and web pages. They are transported in the many parts of many protocols, such as both the control commands and the RFC 2822 body parts of SMTP, and the headers and the body content in HTTP. It is important to remember that host names appear both in host name slots and in the content that is passed over protocols.

In protocols and document formats that define how to handle specification or negotiation of charsets, IDN host name labels can be encoded in any charset allowed by the protocol or document format. If a protocol or document format only allows one charset, IDN host name labels MUST be given in that charset. In any place where a protocol or document format allows transmission of the characters in IDN host name labels, IDN host name labels SHOULD be transmitted using whatever character encoding and escape mechanism that the protocol or document format uses at that place.

All protocols that have generic host name slots already have the capacity for handling host names in the ASCII charset. Thus, IDN host name labels that have been processed with the ToASCII operation can inherently be handled by those protocols.

6.2 Applications and resolver libraries

Applications normally use functions in the operating system when they resolv DNS queries. Those functions in the operating system are often called "the resolver library", and the applications communicate with the resolver libraries through a programming interface (API).

Because these resolver libraries today expect only hostnames in ASCII, applications MUST prepare name parts that are passed to the resolver library using the ToASCII operation. Internationalized labels receiver from the resolver library will always be in ACE form.

IDNA-aware applications MUST be able to work with both non-internationalized host name labels (those that conform to [STD13] and [STD3]) and internationalized host name labels.
It is expected that new versions of the resolver libraries in the future will be able to accept domain names in other formats than ASCII, and application developers might one day pass not only domain names in Unicode, but also in local script to a new API for the resolver libraries in the operating system.

6.3 DNS servers

An operating system might have a set of libraries for performing the ToASCII operation. The input to such a library might be in one or more charsets that are used in applications (UTF-8 and UTF-16 are likely candidates for almost any operating system, and script-specific charsets are likely for localized operating systems).

DNS servers MUST use the ACE format for internationalized host labels. All internationalized names stored in DNS servers MUST be valid names that have been processed with the ToASCII operation.

If a signalling system which makes negotiation possible between old and new DNS clients and servers is standardized in the future, the encoding of the query in the DNS protocol itself can be changed from ACE to something else, such as UTF-8. The question whether or not this should be used is, however, a separate problem and is not discussed in this memo.

6.4 Avoiding exposing users to the raw ACE encoding

All applications that might show the user a host name that was received from a gethostbyaddr or other such lookup SHOULD update as soon as possible in order to prevent users from seeing the ACE. However, this is not considered a big problem because so few applications show this type of resolution to users.

If an application decodes an ACE name using ToUnicode but cannot show all of the characters in the decoded name, such as if the name contains characters that the output system cannot display, the application SHOULD show the name in ACE format instead of displaying the name with the replacement character (U+FFFD). This is to make it easier for the user to transfer the name correctly to other programs. Programs that by default show the ACE form when they cannot show all the characters in a name label SHOULD also have a mechanism to show the name that is produced by the ToUnicode operation with as many characters as possible and replacement characters in the positions where characters cannot be displayed. In many cases, the application doesn’t know exactly what the underlying rendering engine can or cannot display.

In addition to the condition above, if an application receives an ACE host name after performing the ToUnicode operation, meaning that the name was not properly prepared with ToASCII (for example, if it has illegal characters in it), the application MUST show the name in ACE format because the ToUnicode operation never fails, but returns the original input if errors are detected at any step.

6.5 Bidirectional text in host names

The display of host names that contain bidirectional text is not covered in this document. It may be covered in a future version of this document, or may be covered in a different document.

For developers interested in displaying host names that have
bidirectional text, the Unicode standard has an extensive discussion of how to deal with reorder glyphs for display when dealing with bidirectional text such as Arabic or Hebrew. See [UAX9] for more information. In particular, all Unicode text is stored in logical order.

6.6 DNSSEC authentication of IDNA hostnames

DNS Security [DNSSEC] is a method for supplying cryptographic verification information along with DNS messages. Public Key Cryptography is used in conjunction with digital signatures to provide a means for a requester of domain information to authenticate the source of the data. This ensures that it can be traced back to a trusted source, either directly, or via a chain of trust linking the source of the information to the top of the DNS hierarchy.

IDNA specifies that all internationalized host names stored in DNS servers must be valid names processed with the ToASCII operation. This processing must be complete prior to a zone being signed by the private key for that zone. Because of this ordering, it is important to recognize that DNSSEC authenticates the ACE-encoded resource, not the internationalized hostname or the mapping between that hostname and its ACE-encoding form. The canonical name, in other words, is the output of ToASCII. In the presence of DNSSEC, this is the name that MUST be signed in the zone and MUST be validated against. It also SHOULD be used for other name comparisons, such as when a browser wants to indicate that a URL has been previously visited.

One consequence of this for sites deploying IDNA in the presence of DNSSEC is that any special purpose proxies or forwarders used to transform user input into IDNA hostnames must be earlier in the resolution flow than DNSSEC authenticating nameservers for DNSSEC to work.

7. Name Server Considerations

Internationalized host name data in zone files (as specified by section 5 of RFC 1035) MUST be processed with ToASCII before it is entered in the zone files.

It is imperative that there be only one ASCII encoding for a particular host name. ACE is an encoding for host name labels that use non-ASCII characters. Thus, a primary master name server MUST NOT contain an ACE-encoded label that decodes to an ASCII label. The ToASCII operation assures that no such names are ever output from the operation.

Name servers MUST NOT have any records with host names that contain internationalized name labels unless those name labels have been prepared with the ToASCII operation. If names that are not processed by ToASCII are passed to an application, it will result in unpredictable behavior. Note that [NAMEPREP] describes how to handle versioning of unallocated codepoints.

8. Root Server Considerations

Because there are no changes to the DNS protocols, adopting this protocol has no effect on the DNS root servers.

9. 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.

This memo describes an algorithm which encodes characters that are not valid according to STD3 and STD13 into octet values that are valid. No security issues such as string length increases or new allowed values are introduced by the encoding process or the use of these encoded values, apart from those introduced by the ACE encoding 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 host name.

Because this document normatively refers to [NAMEPREP], it includes the security considerations from that document as well.

A. References

[AMC-ACE-Z] Adam Costello, "AMC-ACE-Z version 0.3.1", draft-ietf-idn-amc-ace-z.


[STD3] Bob Braden, "Requirements for Internet Hosts -- Communication Layers" (RFC 1122) and "Requirements for Internet Hosts -- Application and Support" (RFC 1123), STD 3, October 1989.


B. Design philosophy

Many proposals for IDN protocols have required that DNS servers be updated to handle internationalized host names. Because of this, a person who wanted to use an internationalized host name would have to be sure that their request went to a DNS server that had been updated for IDN. Further, that server could send queries only to other servers that had been updated for IDN, because the queries contain new protocol elements to differentiate IDN name labels from current host labels. In
addition, these proposals require that resolvers be updated to use the new protocols, and in most cases the applications would need to be updated as well.

These proposals would require changes to the application protocols that use host names as protocol elements, because of the assumptions and requirements made in those protocols about the characters that have always been used for host names, and the encoding of those characters. Other proposals for IDN protocols do not require changes to DNS servers but still require changes to most application protocols to handle the new names.

Updating all (or even a significant percentage) of the existing servers in the world will be difficult, to say the least. Updating applications, application gateways, and clients to handle changes to the application protocols is also daunting. Because of this, we have designed a protocol that requires no updating of any name servers. IDNA still requires the updating of applications, but only for input and display of names, not for changes to the protocols. Once users have updated the applications, they can immediately start using internationalized host names. The cost of implementing IDN may thus be much lower, and the speed of implementation could be much higher.

C. Authors’ Addresses

Patrik Faltstrom
Cisco Systems
Arstaangsvagen 31 J
S-117 43 Stockholm Sweden
paf@cisco.com

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

Adam M. Costello
University of California, Berkeley
idna-spec.amc @ nicemice.net