

Localized labels for testing IDNs

Cary Karp

Swedish Museum of Natural History

Abstract: Significant attention is currently focused on the technical and policy bases for the introduction of top-level domains with labels derived from character strings with non-ASCII components, as well as on the growing appearance of such labels in lower-level registries. Technical tests are underway in several contexts and this draft is intended to facilitate the comparability of their results. It describes two procedures for generating Punycode strings of arbitrary length that decode to typographically plausible sequences of Unicode characters in any desired script, without requiring particular understanding of unfamiliar writing systems. These strings are intended for testing the response of software applications to encoded sequences of ASCII characters of all possible lengths on all levels of a domain name, and the appearance of the corresponding Unicode strings in the display space. The first approach is primarily intended for laboratory use, and the second for application in the public namespace.

Test requirements

The minimum length of a Punycode string is seven characters, encoding a single non-ASCII character. If current policy constraints on one- and two-character labels are taken to apply to their displayed forms, the minimum length of a stored Punycode string will be nine characters. It should be noted, however, that these length restrictions were established when there was no difference between the stored and displayed forms of a label, and that some registries regard them as attaching to the stored form. It may be expected, nonetheless, that an IDN-aware revision of general policies can place the restriction on the number of displayed characters.

The longest TLD labels currently resolving in the root zone are the six-character .museum and .travel. These are both stored and displayed as ASCII characters but problems have been observed with the response of certain applications to them. However, since similar difficulty has been noted with shorter TLD labels, the problems are more likely to result from failure to recognize the labels as valid TLD designations, than simply from the length of the string. Assuming that requests will be made for new TLDs labeled with dictionary words with display lengths roughly equivalent to .museum and .travel, but written with non-ASCII characters, stored labels of twelve characters and longer will not be uncommon. If, as may also be expected, such things as the names of countries appear in full native representations, the lengths of the stored strings may be significantly greater. This suggests that stored strings of up to the maximum permitted length of 63 characters require evaluation. Even if there is no reason to expect that DNS resolvers will be taxed by the appearance of TLD labels of extreme length, one of the purposes of the technical testing is to identify unanticipated frailty. The response of other widely-deployed applications requires testing, in any case.

In many scripts, the way a character is displayed depends both on its position in a string and on the specific characters adjacent to it. If these shaping properties are to be manifested in the test environment as they are likely to appear in actual registered names, a test string cannot simply be a sequence of randomly selected characters. It can, however, be derived from a word taken from a dictionary of a language written with that script. If an online dictionary is available, its use will ease the determination of the requisite Unicode codepoints and avoid need for the manual transcription of unfamiliar scripts.

A distinction is made between conditions that pertain to laboratory testing in transient namespaces, and those that attach to tests conducted in the public namespace. The former is served by the generation of Punycode strings of varying length, with display-side considerations relating solely to script. The latter, however, involves explicit linguistic considerations. Any label that is entered into the root zone of the DNS for the purposes of IDN testing will be categorically barred from subsequent delegation as a production domain. It is therefore also advisable to use a test term that would be unlikely to appear in that context, or is already restricted from such use. To reduce the potential for difficulty to an absolute minimum, a single word is therefore being recommended for all comparable test purposes.

The approach to creating test strings for use in private namespaces will be illustrated by deriving a non-lexical sequence from the word “hippopotamus” (a term likely to be found both in bilingual desk dictionaries and in corresponding online resources). To obviate any conceivable residual concern about rendering it inviable for subsequent candidacy for encoding into a production TLD label, a numerical sequence will be embedded in it. This is taken from the abbreviation “i18n” for “internationalization”, and is based (but not dependent) on the assumption that no TLD labels will have numerical components. No harm is likely to be done if a resulting string is unrecognizable in the language from which it was derived. The purpose is to generate typographically plausible sequences of characters in a variety of scripts, with no further semantic value or linguistic correctness being necessary (or even desirable). A purely lexical alternative will also be described for application in the public namespace.

Generating test labels

The effect of a TLD label that decodes to a string consisting of a single non-ASCII character (the minimum case) can be determined with the 17-character,

xn--flod18hst-12a

which is generated by the Swedish word *flodhäst* (a vernacular designation for the nearly extinct Swedish Forest Hippopotamus, as well as the non-indigenous hippopotamus species), with the interposed two digits,

flod18häst

This can simply be repeated to yield a Punycode string of arbitrary length, for example, the 27-character,

xn--flod18hstflod18hst-rtbj

from

flod18hästflod18häst

It will also be necessary to test ASCII-only sequences to distinguish between problems resulting solely from string length, and those caused by encoded non-ASCII characters, or otherwise by the Punycode prefix. One example of an ASCII-only string is the 56-character:

hippo18potamushippo18potamushippo18potamushippo18potamus

Total departure from the ASCII realm, using a simple alphabetic script written right-to-left, can be illustrated with the 28-character:

xn--18-rjdb cud0neb9a8ce1ezef

taken from the Yiddish:

היפּוֹטאַם 18 פּוֹטאַם

This has the further advantage of testing embedded characters with opposing directional properties.

The properties of an alphabetic script with sophisticated shaping properties can be demonstrated with the 20-character,

xn--18-dtd1bdi0h3ask

from the Arabic,

فرس 18 النهر

No general assumptions are made about scripts having syllable boundaries or other similarly convenient points for inserting digits into the test string, nor are any assumptions made about appropriate ways to proceed with non-alphabetic scripts. If the numerical device is not applicable, some other means should be applied to make the test sequence useless in the production environment. Such demonstration strings may be devised as appropriate for each desired script. (Note, however, that the string preparation process can place more or less severe constraints on the use of some scripts, as discussed below.)

One example of a label derived from a non-alphabetic script is the 15-character,

xn--18-h31ew85n

from the Kanji,

河18馬

Alternate approach

The procedure described above can be used to generate localized TLD labels for testing in the public namespace. However, that environment differs from the closed laboratory situation in one extremely significant regard – any recognizable linguistic attributes possessed by a string that resolves globally must be rigorously controlled to avoid its being seen as inappropriate by any corresponding speech community. In situations where that cannot be conveniently assured (or where the preceding approach is unnecessarily intricate), one option would be the straightforward use of benign dictionary terms.

A convenient vocabulary is provided by the RFC 2606 list of “Reserved Top Level DNS Names”, which explicitly lists four words that are restricted from autonomous delegation because:

“There is a need for top level domain (TLD) names that can be used for creating names which, without fear of conflicts with current or future actual TLD names in the global DNS, can be used for private testing of existing DNS related code, examples in documentation, DNS related experimentation, invalid DNS names, or other similar uses.”

Of the four names then reserved,

“‘.test’ is recommended for use in testing of current or new DNS related code”

and

“‘.example’ is recommended for use in documentation or as examples.”

The name “example” is also reserved on the second level in the .com, .net, and .org TLDs.

It would be counter to the conditions of RFC 2606 for either “.example” or “.test” to resolve in the root, but no restrictions are placed on lexical equivalents to those terms in other languages. One obvious alternative would therefore be to generate test TLD labels from translations of the word “test” into at least one language using each of the scripts that are represented in the public test. The second-level label in each such TLD could be similarly generated from the word “example”.

As noted above, any test label that is placed in the root zone will be unavailable for subsequent delegation. However, since the words “example” and “test” are already unavailable, similarly barring the

equivalent words in any of the languages figuring in the public test (or perhaps generally in anticipation of future tests) would impose the smallest possible constraint on the production vocabulary. The translated “example” and “test” equivalents can therefore be used freely as IDN test strings in any situation where the intention is for them to be proper dictionary words. All requisite terms will also appear in any bilingual dictionary as discussed above. Examples of strings determined in this manner are उदाहरण.परीक्षा, 싯례.테 슯트, and παράδειγμα.δοκιμή. If longer TLD labels are needed than those generated by a single instance of a translation of “test”, the word can be repeated as required, for example as, тест-тест-тест.

Finally, the utility of the publicly-accessible IDN test TLDs can extend beyond the technical trials specifically intended to precede the introduction of Punycode strings into the production root. It may therefore be worth keeping selected “.test” equivalents in persistent use (pending separated discussion). The utility of any such domain could be further enhanced by permitting the inclusion of second-level domains in addition to the “example.test” equivalents, as specific contributions to the development of IDN are put forward that would clearly benefit from appearing in this manner in the public namespace.

Any such demonstration registry might reasonably be placed under the administration of the IANA, with an appropriate reference group. It would be useful, in any case, to include a shareable list of all test strings generated from “example”, “hippopotamus” and “test”, plus any additional terms used for the purposes described here, in the “IANA Repository of TLD IDN Practices” at,

<http://www.iana.org/assignments/idn/>

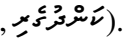
(A seed listing of hippopotami is available at <http://evertype.com/standards/icann/hippo.html>.)

General constraints

For differing reasons, all of the elements of a given writing system that might reasonably be requested for inclusion in a localized domain name may not be available. Some restrictions are inevitable consequences of the domain namespace never having been intended to serve as a vehicle for literary expression. Other limitations result from unanticipated problems and may be eliminated through protocol or policy revision. Converse difficulties have also resulted from excessive latitude in the available repertoire, and some currently viable characters may become unavailable both as IDNA is refined, and as registries adopt more restrictive policies.

Work is in underway in several venues that is intended to clarify and rectify these issues. On first consideration, this might appear to be on a level of detail that is irrelevant to technical trials of the type described above. The agencies conducting such activity may, nonetheless, become engaged in dialog with local communities about specific constraints placed on their languages. This will require some familiarity with the kinds of limitations that are still being addressed, and the ability to assess the degree of transiency of specific issues. Key pending details are therefore reviewed below.

Many symbols that are neither alphanumeric nor ideographic components of a written language, such as line-drawings and pictographic dingbats, are currently permitted in IDNA but are likely to be blocked in a coming revision. Discussions are still being conducted about the extent to which other non-literal and non-numerical characters should be available for inclusion in alphanumeric strings, and about corresponding issues in the ideographic realm. In one sense, this equates to the consideration of permitting punctuation marks in addition to the hyphen. Many scripts use other symbols for purposes roughly parallel to the function of the hyphen in English orthography, but do not recognize the hyphen at all. Despite this, it is unlikely that there will be any general rule about one (or some other small number) of symbols being made available for every such script.

There are, however, situations with specific languages where non-literal adjunct marks can be seen as essential elements of even a skeletal orthography. In addition to the prohibitive criterion of visual confusability with protocol elements, directional properties are also an important factor. IDNA currently requires that a string of characters in a script that is written right-to-left neither begins nor ends with a combining mark. (A string of left-to-right characters may not begin with a combining mark either, but it may end with one.) The clearest example of resulting difficulty that has thus far been noted is with Dhivehi, the official language of Maldives. This is written in the Thaana script (in the Unicode block U+0780...U+07BF), which requires the addition of a combining mark to every base character. A vowel following a consonant is indicated with a combining mark, and special combinations are used to indicate consonants and double vowels in syllable-final position. Every Thaana string thus ends with a combining mark and will be rejected by Stringprep (as illustrated with the Dhivehi word for hippopotamus, ) .

There are reasonable IDN labels derived from other languages written with right-to-left scripts that will be similarly rejected because of final combining characters, and there are also cases with left-to-right scripts where the label-final character cannot be correctly represented. One example of this is the lowercase Greek final sigma “ς”, which is normalized to the initial and medial form “σ” and never displayed at the end of Unicode string that has been decoded from a stored Punycode sequence. This prohibits the correct representation of many names, such as that of the country Cyprus, which can only be incorrectly represented in IDN as “κυπρσσ”. The German Esszet “ß” is similarly irrecoverably normalized to “ss” in the encoding process. (Both the final sigma and Esszet are, however, acceptable input and may appear in the presentation form of a URL, or in offline publication.)

Appendix 1

The discussion of earlier versions of this text has revealed need for clarification about the first phase of the technical trials that ICANN has commenced in the interim. This is focused exclusively on the response of name server and resolver applications to Punycode strings of varying length when they appear as top-level labels. The test strings (listed in Appendix 2) were encoded from scripts selected on the basis of the detail discussed in the body of this text. This selection was made without regard to demographic or other socio-linguistic factors that pertain to any languages written with these scripts.

Although the dictionaries of specific languages were consulted at the outset of the preparation of these test strings, the subsequent process methodically transformed the initial terms into character sequences that are not words. However, since the display forms of the test strings may need to have plausible typographic properties in subsequent phases of the test action (or in parallel action elsewhere), some vestigial word-like appearance remains. The initial laboratory trial, which is currently in progress, does not consider any display properties of either the Unicode or Punycode forms of the strings; it is targeted on resolution behavior only. If the display form of any sequence is discovered to be inappropriate in any unforeseen regard, it will simply be removed from the roster for coming tests.

The prioritization of scripts as they appear in the public namespace in the second phase of the test being coordinated by ICANN is a matter for separate discussion. This will be an instantiation of the fully lexical approach described above. The test strings will be translations of “example.test” determined through an open call for expressions of interest in specific representations for use during the trial. The vetting process for determining which of the submitted “example.test” proposals will then be entered into the root zone of the DNS is in an early stage of consideration, as is the lifespan of the test domains. These decisions may be informed by the outcome of other technical trials in private namespaces, which is the reason for the detailed discussion of string selection methodology in the present document.

Appendix 2

The following Punycode strings are being used for laboratory tests of widely deployed name server and resolver implementations under the conditions discussed above. This list is subject to modification and may be updated in subsequent versions of this document.

```
xn--18-7g4a9f
hippo18potamus
xn--18-xf0jl42g
xn--18-h31ew85n
xn--flod18hst-12a
xn--18-xsdrfd6ex1e
xn--18-dtd1bdi0h3ask
xn--18-28gg3ad5hl2fzb
xn--18-hmf0e1bza7dh8ioagd6n
xn--18-rjdb cud0neb9a8ce1ezef
xn--1818-63dcpd5be6bfqcecfadfad3dl
xn--1818-1goc0bacbac7eg2kh6ci9cj9bk4yla7abl b
xn--181818-qxecc5edd8aee8aebecadeadead0fkill15yam
xn--flod18hstflod18hstflod18hstflod18hstflod18-1iej j j j
hippo18potamushippo18potamushippo18potamushippo18potamus18hippo
```

Acknowledgements

This draft originated in a request by the ICANN President's Advisory Committee on IDN for a list of localized TLD labels for use in a series of impending technical trials. The text was then discussed by the working group that maintains the ICANN Guidelines for the Implementation of IDN, during the course of its work toward rendering that document applicable to the top-level of the DNS. Further valuable commentary was received from an informal group of IETF members considering the need for modification of the IDNA protocol, and other colleagues.

All members of these groups are thanked for their advice, with particular acknowledgment of the contributions of the following people (in alphabetical order):

Harald Alvestrand
Tina Dam
Mohammed El Bashir
Michael Everson
Patrik Fältström
Hiro Hotta
Pat Kane
John Klensin
Ram Mohan

Author's address:

Cary Karp <ck@nrm.museum>
Swedish Museum of Natural History
Frescativägen 40
SE-10405 Stockholm