L2/15-056

Subject: ABNF for domain names in UTS #46
From: Mark Davis
Date: 2015-02-01

I had the following action:

140	A066	Mark Davis	Look at feedback from Anne van Kesteren in L2/14-179 re UTS #46, and report back to UTC with a recommendation whether to do anything, and if so, what.

His feedback was:

> I just wanted to clarify something with regards to the review note at the end of http://www.unicode.org/reports/tr46/proposed.html#Implementation_Notes
>
> What I'd really like to see is a syntax description. E.g. a domain consists of domain labels separated from each other by domain label separators, optionally with a trailing domain label separator. A domain label is a sequence of one or more code points which are one of X, Y, and Z. A domain label separator is one of X, Y, and Z. Alternatively you could express this using ABNF or some kind of grammar.
>
> That is the kind of thing people writing validators or authoring tools are often looking for. And often web developers as well. They don't want to have to put some input they made up through a series of functions before they know whether the input is valid. I guess another way of saying this would be having a declarative description of a domain.
>
> (This is an open issue https://www.w3.org/Bugs/Public/show_bug.cgi?id=25334 for the URL Standard.)

For context, the review note in the draft at the time was:

Review Note: *Should we point to information that answers the following questions?*

1. *Whether to store URLs with domain names in Punycode format or Unicode.*
2. *How to syntactically determine the boundary of a domain name in a URL, given the mappings and conversions of characters.*
3. *How to determine the end of a URL in plain text such as email contents (the so-called "linkification" process).*

It is not possible, in my opinion, to provide any ABNF syntax that would answer definitely **yes** or **no** to the question of whether a string was a valid domain name or not, because of the normalization and mapping involved.

I think what would be possible is to define a "quick-check syntax" for a domain name, somewhat like what we do for normalization. That is, we could probably devise a syntax that would do a reasonable "lower-bounds" check. If the string didn't satisfy the lower-bounds syntax, it would definitely not be a valid domain name. For example, this could be done for checking for characters that were not valid and could never become valid under normalization or mapping.

And it would probably be possible to define the converse syntax: if a string did satisfy the upper-bounds syntax, then it would definitely be a valid domain name. For this check, we'd look for strings that satisfied the normalization-quick-check test, had only valid characters, and had no mappings to characters that wouldn't satisfy the normalization-quick-check test.

Anything that satisfied the lower-bounds syntax check and didn't satisfy the upper-bounds syntax check would be a maybe, and would need the comprehensive check by running http://www.unicode.org/reports/tr46/#ToASCII (with the appropriate processing options).

I think developing such syntax would not really help spec writers; what it would provide is a possible optimization. However, one would have to measure the extra cost of applying these two tests against the cost of ToASCII, taken against the expected probabilities of getting a definitive **yes** or **no** answer over the sample space of possible URLs. It would definitely take some time to develop and test this optimization. So we'd need someone willing to commit the time to research this…

However, I think what would be of value to add to #46 would be a section outlining a recommended approach to #3 from the review note above: the linkification problem. It is a tricky problem to find reasonable endpoints within flowing text, a problem that calls more than anything on knowledge of Unicode, character properties, and natural languages. (And speaking from experience, since I've been looking at this problem, even within the same company you'll find a dozen different API solutions, each giving different results.) However, we should not expect anything like that in the 8.0 timeframe.