```
Subject: Re: Unicode Technical Committee Liaison Statement to IETF on iab-idn-nextsteps
From: Mark Davis <mark.davis@icu-project.org>
Date: Friday, 24 Feb 2006 01.52.32 GMT+02:00
To: paf@cisco.com,
CC:iab@iab.org, john-ietf@jck.com, "rick@unicode.org" <rick@unicode.org>
```

The Unicode Technical Committee has reviewed the document http://www.iab.org/documents/drafts/draft-iab-idn-nextsteps-02.txt.

The UTC strongly supports many of the goals of the document, including especially improving the security of IDNs, and updating the version of Unicode used in NamePrep and StringPrep (since the old version of Unicode they require excludes or hampers many languages). There are, however, a number of areas of concern.

As a general issue, we'd urge closer cooperation between the IAB and the Unicode consortium on the document, so that the character encoding and software internationalization issues can be reviewed by experts in the field, and accurately represented in the document.

The chief area of concern is section 4.3.

```
4.3.  Combining Characters and Character Components

   One thing that increases IDNA complexity and the need for
   normalization is that combining characters are permitted.  Without
   them, complexity might be reduced enough to permit more easy
   transitions to new versions.  The community should consider whether
   combining characters should be prohibited entirely from IDNs.  A
   consequence of this, of course, is that each new language or script
   would require that all of its characters have Unicode assignments to
   specific, precomposed, code points, a model that the Unicode
   Consortium has rejected for Roman-based scripts.  For non-Roman
   scripts, it seems to be the Unicode trend to define such code points.
   At some level, telling the users and proponents of scripts that, at
   present, require composing characters to work the issues out with the
   Unicode Consortium in a way that severely constrains the need for
   those characters seems only appropriate.  The IAB and the IETF should
   examine whether it is appropriate to press the Unicode Consortium to
   revise these policies or otherwise to recommend actions that would
   reduce the need for normalization and the related complexities.
```

The descriptions and recommendations in this section are simply not feasable. They do not recognize the fundamental importance of combining marks as an integral component of a great many scripts, nor do they recognize the fundamental need for compatibility that is required of the Unicode Standard. Asking for combining characters to be removed is akin to asking English vowels to be removed, and all possible syllables to be encoded instead. There are, as well, a number of purely factual errors. For example, "it seems to be the Unicode trend to define such code points" is simply incorrect. This section serves no purpose but to betray a basic lack of understanding of scripts; it needs to be removed entirely.

A second area of major concern is Section 2.2.3.

```
2.2.3.  Normalization and Character Mappings

   Unicode contains several different models for representing
   characters.  The Chinese (Han)-derived characters of the "CJK"
   languages are "unified", i.e., characters with common derivation and
   similar appearances are assigned to the same code point.  European
   characters derived from a Greek-Roman base are separated into
   separate code blocks for "Latin", Greek and Cyrillic even when
   individual characters are identical in both form and semantics.
   Separate code points based on font differences alone are generally
   prohibited, but a large number of characters for "mathematical" use
   have been assigned separate code points even though they differ from
   base ASCII characters only by font attributes such as "script",
   "bold", or "italic".  Some characters that often appear together are
   treated as typographical digraphs with specific code points assigned
   to the combination, others require that the two-character sequences
   be used, and still others are available in both forms.  Some Roman-
   based letters that were developed as decorated variations on the
   basic Latin letter collection (e.g., by addition of diacritical
   marks) are assigned code points as individual characters, others must
   be built up as two (or more) character sequences using "composing
   characters".
```

This section betrays a lack of understanding of the fundamental differences between Han characters and the scripts Latin, Greek, and Cyrillic.

```
   Many of these differences result from the desire to maintain backward
   compatibility while the standard evolved historically, and are hence
   understandable.  However, the DNS requires precise knowledge of which
   codes and code sequences represent the same character and which ones
   do not.  Limiting the potential difficulties with confusable
   characters (see Section 2.2.6) requires even more knowledge of which
   characters might look alike in some fonts but not in others.  These
   variations make it difficult or impossible to apply a single set of
   rules to all of Unicode.  Instead, more or less complex mapping
   tables, defined on a character by character basis, are required to
   "normalize" different representations of the same character to a
   single form so that matching is possible.
```

The Unicode consortium *does* supply a precise mechanism for determining when two strings represent the same underlying abstract characters. These do supply a single set of rules to all of Unicode, based on a set of data that is in the Unicode Character Database.

This paragraph also conflates the confusable issue with character equivalence. These are separate issues: there are great many instances where characters are confusable where they are not at all equivalent (such as zero and the letter O).

```
   ... The fact
   that most or all scripts included in Unicode have been initially
   incorporated by copying an existing standard more or less intact has
   impact on the optimization of these algorithms and on forward
   compatibility.  Even if the language is known and language-specific
   rules can be defined, dependencies on the language do not disappear.
   Any canonicalization operations that depend on more than short
   sequences of text is not possible to do without context.  DNS lookups
   and many other operations do not have a way to capture and utilize
   the language or other information that would be needed to provide
   that context.
```

First, it is neither "most" nor "all". Very few scripts, proportionately, have been incorporated by copying an existing standard. Second, "Any canonicalization operations that depend on more than short sequences of text is not possible to do without context...." is difficult to make sense of. One would have to explain the sense of "canonicalization" that is being discussed. It could be as trivial as "language-based canonicalization is impossible without language information", which is true, but above the document argues against using language-based equivalencies on a global basis (and for very good reason!)

===

Other areas of concern:

```
   (more properly "Roman", see below)
```

The common modern practice in the naming of the script is to use the term "Latin", not "Roman". Whether or not one thinks that should not have been the case, insisting on older terms is pointless, and not germain to the purpose of the document.

```
   When writing or typing the label (or word), a script must be selected
   and a charset must be picked for use with that script.
```

This is confusing charset, keyboard and script. Saying "a script must be selected" is *neither* true from the user's perspective, nor does it at all match the implementation pipeline from keypress to storage of a label. What may have been confusing for the authors is that sometimes keyboards that are listed for selection are sorted by script; that does not, however, mean that a "script is selected".

The proper word, if more substantial changes are not made to the wording, would be "a keyboard must be selected". (Even that is a quite odd, since it implies that that is done each time a user types a label.)

```
   If that charset, or the local charset being used by the relevant
   operating system or application software, is not Unicode, a further
   conversion must be performed to produce Unicode.  How often this is
   an issue depends on estimates of how widely Unicode is deployed as
   the native character set for hardware, operating systems, and
   applications.  Those estimates differ widely, with some Unicode
   advocates claiming that it is used in the vast majority of systems
   and applications today.  Others are more skeptical, pointing out
   that:

   o  ISO 8859 versions [ISO.8859.1992] and even national variations of
      ISO 646 [ISO.646.1991] are still widely used in parts of Europe;
   o  code-table switching methods, typically based on the techniques of
      ISO 2022 [ISO.2022.1986] are still in general use in many parts of
      the world, especially in Japan with Shift-JIS and its variations;
   o  that computing, systems, and communications in China tend to use
      one or more of the national "GB" standards rather than native
      Unicode;
   o  and so on.

   Not all charsets define their characters in the same way and not all
   pre-existing coding systems were incorporated into Unicode without
   changes.  Sometimes local distinctions were made that Unicode does
   not make or vice versa.  Consequently, conversion from other systems
   to Unicode may potentially lose information.
```

Most of this section is unnecessary and the thrust of it is misleading. The only issue is "local distinctions" are lost when converting to Unicode; that doesn't happen when converting from any of the examples listed. This passage implies that there are significant problems in mapping to Unicode in doing IDN, and there simply aren't.

```
   ... Worse, one needs to be reasonably
   familiar with a script and how it is used to understand how much
   characters can reasonably vary as the result of artistic fonts and
   typography.  For example, there are a few fonts for Latin characters
   that are sufficiently highly ornamented that an observer might easily
   confuse some of the characters with characters in Thai script.
```

The confusion of Latin with Thai is a red herring. It would take an exceedingly contrived scenario for it to present a problem. There are plenty of realistic scenarios involving confusables across, say, Latin and Cyrillic.

```
   ... IDNA
   prohibits these mixed-directional (or bidirectional) strings in IDN
   labels, but the prohibition causes other problems such as the
   rejection of some otherwise linguistically and culturally sensible
   strings.  As Unicode and conventions for handling so-called
   bidirectional ("BIDI") strings evolve, the prohibition in IDNA should
   be reviewed and reevaluated.
```

Deviating from the practices already built into IRI would be a mistake. As the document recognizes above, it cannot be a goal to represent all possible "linguistically and culturally sensible strings" in IDNs. The restrictions on BIDI are ones that have achieved broad consensus as the minimal ones to help avoid some fairly serious security issues.

```
4.1.2.  Elimination of word-separation punctuation
   ... We might even
   consider banning use of the hyphen itself in non-ASCII strings or,
   less restrictively, strings that contained non-Roman characters.
```

This section is not well motivated. The authors need to justify why such characters represent a problem (and one of such a serious nature that hyphens should be disallowed).