Internationalized Addresses in XMPP
(draft-saintandre-xmpp-i18n-03)

Peter Saint-André
PRECIS WG / XMPP WG
IETF 80, Praha, Česká Republika
XMPP Input

• These slides describe possible input of the XMPP WG to the PRECIS WG

• We do not yet have consensus about these proposals in the XMPP WG

• The intent is to start discussion, not end it!
Unicode Recap (1)

• Every character is a "code point"

• Characters have properties, e.g.:
  • letter, number, symbol, etc.
  • uppercase vs. lowercase vs. titlecase
  • modifiers (e.g., accent marks)
  • left-to-right vs. right-to-left
Unicode Recap (2)

- We decide how to handle characters based on their properties.
- A character can be *equivalent* to another character or a sequence of characters.
- Things like Å and ç are "composite characters" (humans like them).
Unicode Recap (3)

• Two kinds of equivalence

• Canonical: "this character is the standard for that one" (e.g., Å ≡ Å or ç ≡ c + \,)

• Compatible: "this character suffers with that one" (e.g., IV ≈ I + V or ß ≈ s)
Unicode Recap (4)

- *Decomposition* analyzes a character into its component units
- Two kinds of decomposition: canonical and compatible
- Order matters (e.g., ḷ ≡ ω + ′ + ~ + ‼)
Unicode Recap (5)

• *Normalization* removes alternate representations of equivalent sequences so we can convert the data into a form that can be compared for equivalence

• Normalization can involve both decomposition and recomposition, and both canonical and compatibility rules
# Unicode Recap (6)

<table>
<thead>
<tr>
<th></th>
<th>Canon Decomp</th>
<th>Compat Decomp</th>
<th>Canon Recomp</th>
<th>Compat Recomp</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFD</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NFKD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>NFC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NFKC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Friday, March 25, 2011
As we know, IDNA2008 moved away from stringprep for domain names

Other technologies want to move as well (for Unicode agility and other reasons)

PRECIS WG is working on a replacement for use by other stringprep "customers"

XMPP WG to provide input to PRECIS
PREMIS Recap (2)

• Stringprep provided:
  • Mappings (e.g., spaces, prohibited characters, case folding)
  • Normalization (typically NFKC)
  • Handling of right-to-left scripts
• PREMIS to provide similar "services"
• Pursue inclusion approach
• Define common string classes
• Enable subclassing of string classes
• Define processing rules for each class based on Unicode properties
• Specify mapping rules (probably)
String Classes

- Four string classes of interest in XMPP:
  - "Nameythings" for localparts
  - "Stringythings" for resourceparts
  - "Wordythings" for passwords (cf. SASL)
  - "Domaineythings" for domainparts (in IDNA, but we need common mapping)
Nameythings (1)

- Purpose: usernames, chatroom names, etc.
- Can be subclassed by application protocols (e.g., to prohibit additional codepoints)
- In XMPP, will be used as base class for localpart of JID (thus replacing Nodeprep)
Nameythings (2)

- Disallowed:
  - Space characters (GeneralCategory = Zs)
  - Control characters (GC = Cc)
  - Any character that has a compatibility equivalent (as in IDNA2008)
- OPEN ISSUE: Full-width / half-width codepoints in Asian scripts
Nameythings (3)

- Protocol Valid:
  - All other 7-bit ASCII characters (even if GeneralCategory otherwise disallowed)
  - Letters, digits, punctuation, symbols
  - OPEN ISSUE: Do symbols really need to be protocol-valid? (e.g., "the👑", "i♥ny")
Nameythings (4)

• Fold uppercase and titlecase codepoints to their lowercase equivalents

• OPEN ISSUE: Right-to-left codepoints

(note: the "Bidi Rule" from RFC 5893 is more complex than we need because nameythings do not have internal structure)
Stringythings

- As with nameythings except:
  - Spaces are protocol-valid
  - Characters with compatibility equivalents are protocol-valid
  - Symbols are (certainly) protocol-valid
  - No case folding
Wordythings

• As with nameythings except:

  • Characters with compability equivalents are protocol-valid
  • Symbols are (certainly) protocol-valid
  • No case folding
Domaineythings

- Use what's defined in IDNA2008
- But, might need common mapping for use over the wire in XMPP and perhaps other application protocols (e.g., apply case folding and NFD)
Why NFD?

• Simplest normalization form
• We can simply disallow characters requiring compatibility decomposition
• We don't need recomposed characters on the wire or in storage
• Client-side font rendering can handle recomposition if needed
Subclassing

• Do we really need to subclass the base classes?

• Are the string classes really subclasses of some "Ur-class"?

• Flexibility might introduce interoperability challenges across application protocols (e.g., email account vs. IM account)
PRECIS Open Issues

- Which string classes?
- Benefits and hazards of subclassing
- Full-width / half-width code points
- Right-to-left outside IDNA
- Normalization form(s)
- Mapping recommendations
XMPP Open Issues

- Clarify error handling
- Specify client and server responsibilities
- Create list of all JID / JID-part slots
- Define "registrar" policies for servers?
- Create UI guidelines for clients?
- Formulate migration plan