Current work in Internationalization Around the IETF

John Klensin

General i18n character issues

- Making Unicode the "new ASCII“
- Long-standing interoperability tension:
  - Standard on the wire, conversion at endpoints
  - Negotiation with servers for client-desired charsets

Unicode in Practice

- Except in
  - Completely free text (e.g., no matching or sorting)
  - Markup

  … Issues of normalization and exclusions
- In the general case,
  - can’t code a character in isolation
  - What is “A” ?
- Standard gets bigger
  - ASCII was a closed set – no characters added

Unicode Encodings

- With ASCII, one encoding
  - Seven bits, in eight-bit field, leading zero bit (RFC20)
  - Exactly one way to code each character
- Unicode
  - Coding
    - UTF-32 (UCS-4)
    - UTF-16
    - UTF-8
    - Any big-endian or little-endian in principle
  - Many characters can be coded in different ways

Patrik’s Slides Go Here

Target
An Internationalized Internet

- Important to get things right
  - Reasons for ASCII focus decades ago
    - No choice, not no interest
    - Protocol design versus UI design
- What would it look like if we started today
    - Protocol elements *still* in ASCII
    - Still in need of normalization and exclusion in many contexts

Many Foundation Pieces

- Some done or nearly done (we hope)
  - Language Tag work: LTRU (RFC4645 etc.)
  - Comparators, collation, and registry (RFC4790)
  - UTF-8 definition (RFC3629)
- Some being reexamined after experience
  - IDNA (RFC3490 etc.)

Some Actual Applications Work in Various Stages

- Protocols
  - Email Address & Header Internationalization
  - IMAP Extensions
  - SASL & Certificate Work
- Pain level example
  - Email addresses are used as identifiers all over the net
  - Many applications and databases can’t get existing ones right (RFC3696) – real opportunities with internationalized ones.

More Foundation Work

- Unicode in ASCII Contexts
  - Escapes in old protocols
    - draft-klensin-unicode-escapes
- Standardizing a text stream form
  - Protocols using unstructured text data
    - Historically: telnet, ftp, whois, …
    - But even the text/plain media type
  - Parallel to “net ASCII” / NVT ASCII
    - draft-klensin-net-utf8

IDNA: A 30 Second Summary

- Non-ASCII strings converted to coded form (“Punycode”) by applications.
- Special prefix “xn—” to distinguish from conventional domain names
- Many Unicode characters mapped to others (mappings are one-way)
- Mappings and procedure tied to Unicode 3.2: no upgrade plan
- Unicode sequences are normalized to remove some representation differences
- Result looks to DNS like a host (LDH)-type name – *No* DNS changes.

IDN Issues
More on the IAB Report (RFC4690)

- Problems that need solving and might be solvable
  - Too much confusion about what really happens
  - Versions of Unicode
  - Character confusion
    - Largely a registration problem
    - Can't rely completely on registries
  - Some Unicode non-optimality for IDN use

Proposed New Work

- Highlights
  - Terminology
  - Isolating UI Issues from Protocol
  - Unicode Version-agility
  - Inclusion and Reduced Character Collection
  - Technical Fixes

  …Coming back to most of these…
- No Changes to Stringprep that would affect other protocols
- Overview in draft-klensin-idnabis-issues

IDNA: New Terminology

- What string does “Punycode” designate? What are its properties?
  - With or without prefix?
  - Valid or not?
- So
  - U-labels
  - A-labels
  - LDH-labels

Unicode Versions

- INDA 2003 linked to Unicode 3.2
- Applications use libraries to do the work
  - Libraries change with operating system and language updates
  - Application often can't tell version
  - So "defined for one version of Unicode" is meaningless in practice
- Solution: Make protocol insensitive to Unicode versions – "Version-agile"

IDNA: Localization and UI

- Users don't need i18n except for I10n
- Key UI issue is proper localization for
  - Culture -- Fonts
  - Language -- Presentation & Display
- Can't be reflected in DNS
- Hence
  - Less mapping in protocol
  - U-label ↔ A-label
  - Standardizing forms in IRI to U-label

IDNA: New Tables

- Inclusion, not Exclusion
  - Need a reason
  - “Want to” is not a reason
  - DNS integrity and ability to parse names in context are key goals
- Table Model
  - Yes
  - No (“language characters” only)
  - Pending (probably yes)
  - Pending (probably no or much later)

New IDN Tables
Symbols and Punctuation

- All excluded
- Parallels Hostname Rules
- Avoids parsing problems when embedded in other protocol strings (e.g., URIs/ IRIs)
- Not rational to make decisions one character at a time

New IDN Tables
The Pending→ Yes Transition

- "Language Characters" only
  - Characters that can be used to write words
  - Of course, DNS labels don't need to be words
- Requires a user community for script
- Special presentation issues must be sorted out.
  - Position-dependent presentation
  - The "zero-width" objects

IDNAbis
Changes in Practice

- Character → Character mappings become UI responsibility
  - Most reasonable ones won't change
- More Characters
  - Larger number permitted in registered strings
  - Better BIDI treatment
- No Unicode version restriction
- All easier to understand and explain

Thanks To

- The IDNAbis team
  - Harald Alvestrand
  - Patrik Fältström
  - Tina Dam
  - Cary Karp
- Other contributors to presentations
  - Leslie DAIGLE
  - Mirjam KÜHNE
  - LEE XiaoDong
  - Ted HARDIE
  - Hichem MAALAOUI
  - Pete RESNICK

Discussion Lists

- General: discuss@apps.ietf.org
  - net-utf8
  - unicode-escapes
- IDNAbis: idna-update@alvestrand.no
- WG Documents
  - Obvious WG mailing lists – see charters