PRECIS Framework

(draft-blanchet-precis-framework &
draft-saintandre-xmpp-6122bis)

Peter Saint-André
PRECIS WG
IETF 81, Québec City

Goals

- Build a usable framework for preparation and comparison of i18n strings
- Provide Unicode agility
- Define common string classes
- Enable protocols to subclass if needed
- Leave "mapping" (case, normalization, bidi) up to using protocols

2

Framework

- Same principles as IDNA2008
  - Algorithmic categories based on properties of Unicode code points (big character tables at IANA are informative, not normative)
  - Agility w.r.t. Unicode versions
- Reuse IDNA "categories" where possible
- Define new categories where necessary

3

String Classes

- NameClass – for usernames, filenames, and other "mnemonics"
- SecretClass – for passwords, passphrases, and other secrets
- FreeClass – for nicknames, display names, and other free-form strings

4

NameClass: PVALID

- Letters
- Digits
- Any character in 7-bit ASCII range (even if GeneralCategory otherwise disallowed)

5

NameClass: DISALLOWED

- Control characters
- Space characters
- Symbol characters
- Punctuation characters
- Any character with a compatibility equivalent

6

NameClass: Mapping

- Case: up to application protocol
- Normalization: up to application protocol, but NFC recommended (cf. RFC 5198)
- Bidi: up to application protocol

7

SecretClass: PVALID

- Letters
- Digits
- Any character with a compatibility equivalent
- Any character in 7-bit ASCII range (even if GeneralCategory otherwise disallowed)

8

SecretClass: DISALLOWED

- Control characters
- Space characters
- Symbol characters
- Punctuation characters

9

SecretClass: Mapping

- Case: up to application protocol, but case preservation recommended to maximize entropy
- Normalization: up to application protocol, but NFC recommended (cf. RFC 5198)
- Bidi: up to application protocol

10

FreeClass: PVALID

- Letters
- Digits
- Space characters
- Symbol characters
- Punctuation characters
- Any character with a compatibility equivalent

11

FreeClass: DISALLOWED

- Control characters

12

FreeClass: Mapping

- Case: up to application protocol
- Normalization: up to application protocol, but NFC recommended (cf. RFC 5198)
- Bidi: up to application protocol

13

Case Study: XMPP

- Two identifiers: localpart and resourcepart
- Localpart subclasses NameClass to prohibit special characters, case folding to lowercase, NFC (?), any RTL character makes entire string RTL
- Resourcepart uses FreeClass without subclassing, case preserved, NFC (?), any RTL character makes entire string RTL
- Use of PRECIS is easy, the hard part is migration

14

Open Issues

- Have we defined the right string classes?
- Have we defined them correctly?
- What are the benefits and hazards of subclassing?
- Do we need special handling for full-width and half-width code points in certain Asian scripts?
- Should we make mapping recommendations?
- Do we need BackwardCompatible lists for future breaking changes in Unicode property?

15