Internationalised Domain Names in Europe

Kim Davies «*kim@centr.org*»
Council of European National Top Level Domain Registries

Amsterdam, Netherlands
29 January, 2004

What are IDNs?

- Internationalised Domain Names provide an *extra layer* on the DNS to provide characters for the world's languages.
- Allows great repertoire of characters – subset of the Unicode 3.2 set.
- As it is an extra layer, doesn't change the underlying DNS protocol.
  - Tradeoff means core Internet infrastructure doesn't change, but provides other challenges...

The Unicode Spectrum

(A very small selection. See http://www.unicode.org/charts/)

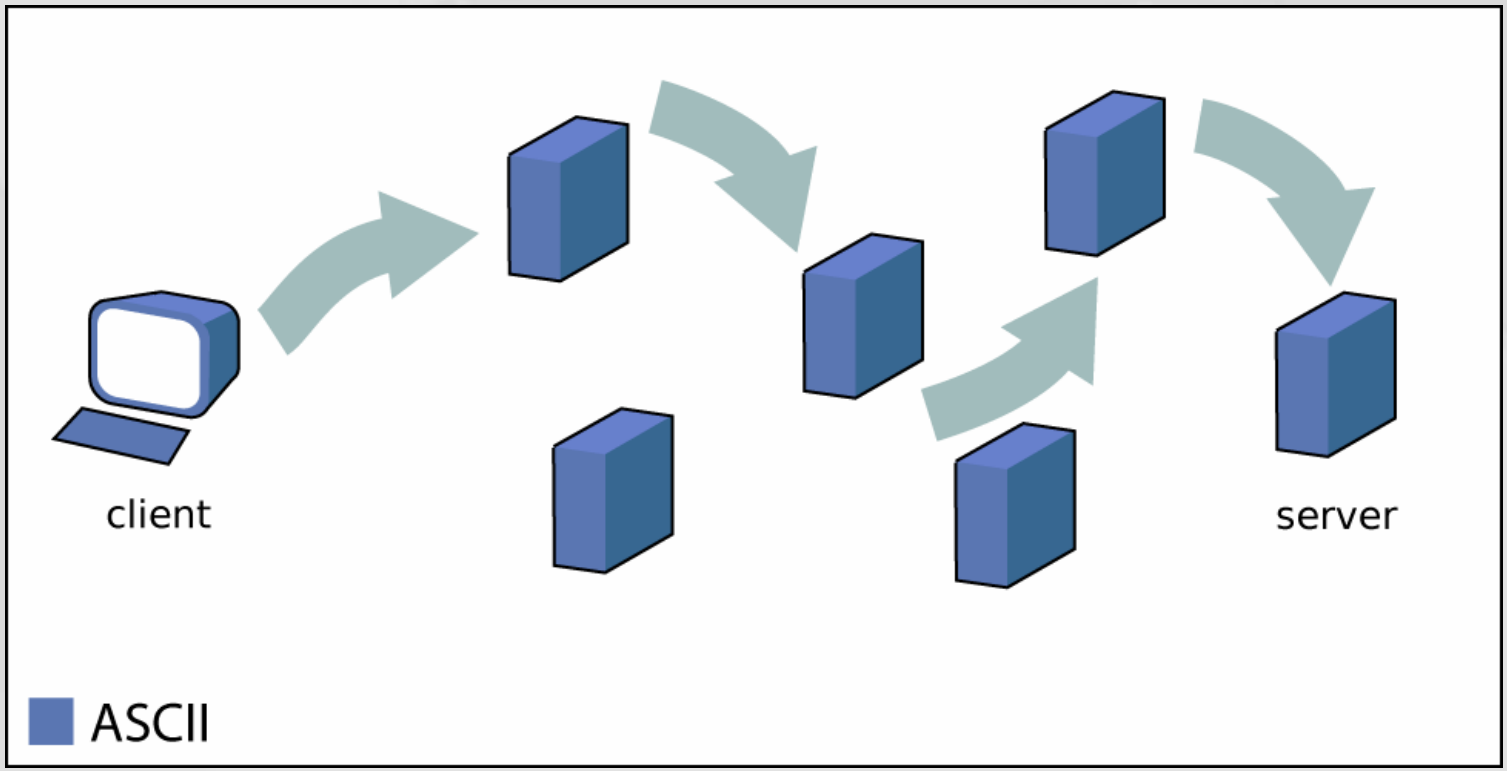
The IDN Layer

- This extra layer converts between two different forms
  - *ASCII domains (LDH i.e. A-Z, 0-9, hyphens)*
  - *Unicode domains*
- Unicode domains and transformed into ASCII domains so they can pass over the existing unchanged Internet.
- Therefore need to be encoded.
  - Encoding scheme called “punycode”.
  - Generically referred to as “ASCII compatible encoding”, or “ACE”

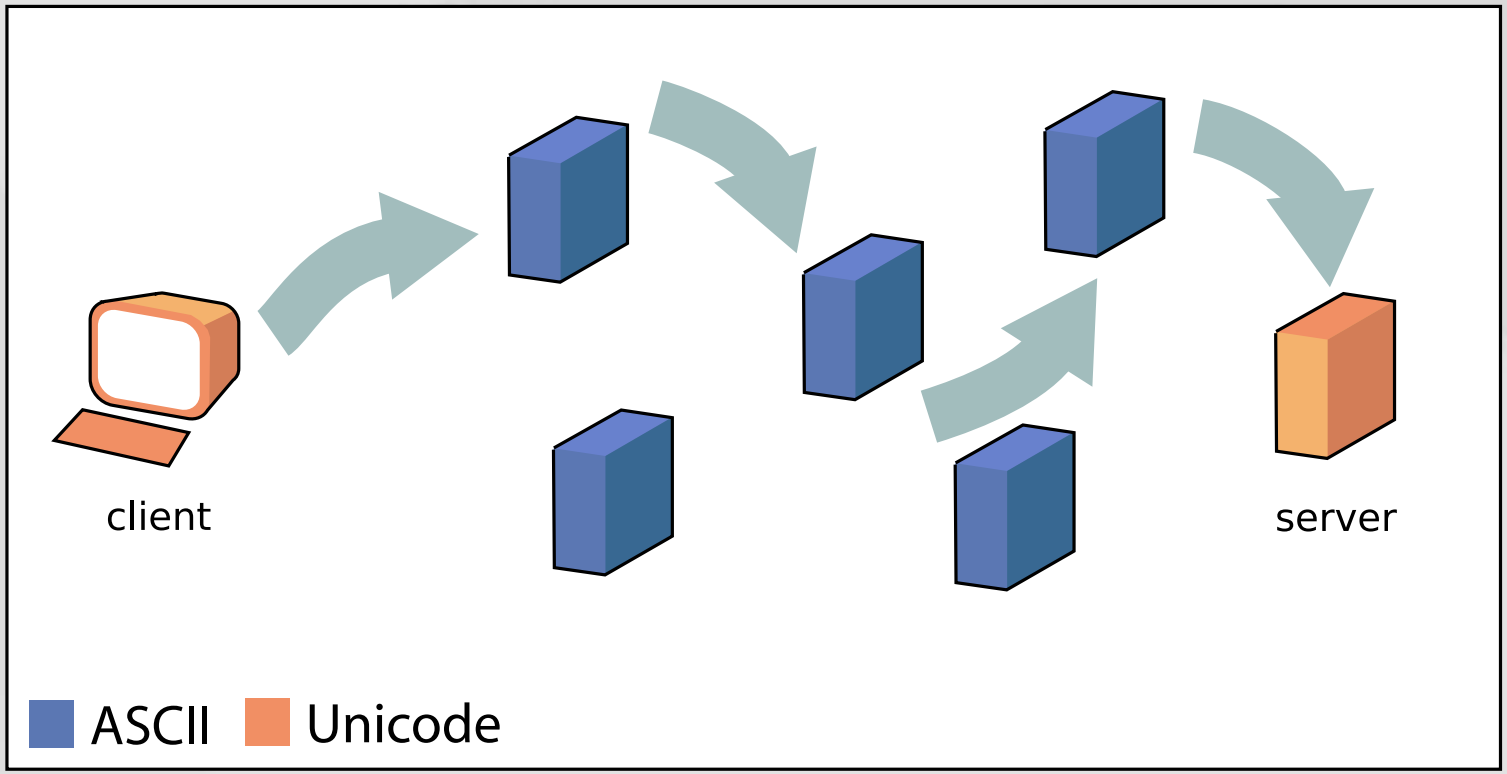
Two forms – where are they?

- There are two forms of IDNs
  - the ASCII form
  - the Unicode form.
- The ASCII encoding scheme always begins with ***xn--*** so distinguishable from other domains.
- The ASCII version is what travels over the Internet. The Unicode form is reserved for the very end – in software, such as web browsers and mail programs.

Without IDNs



With IDNs

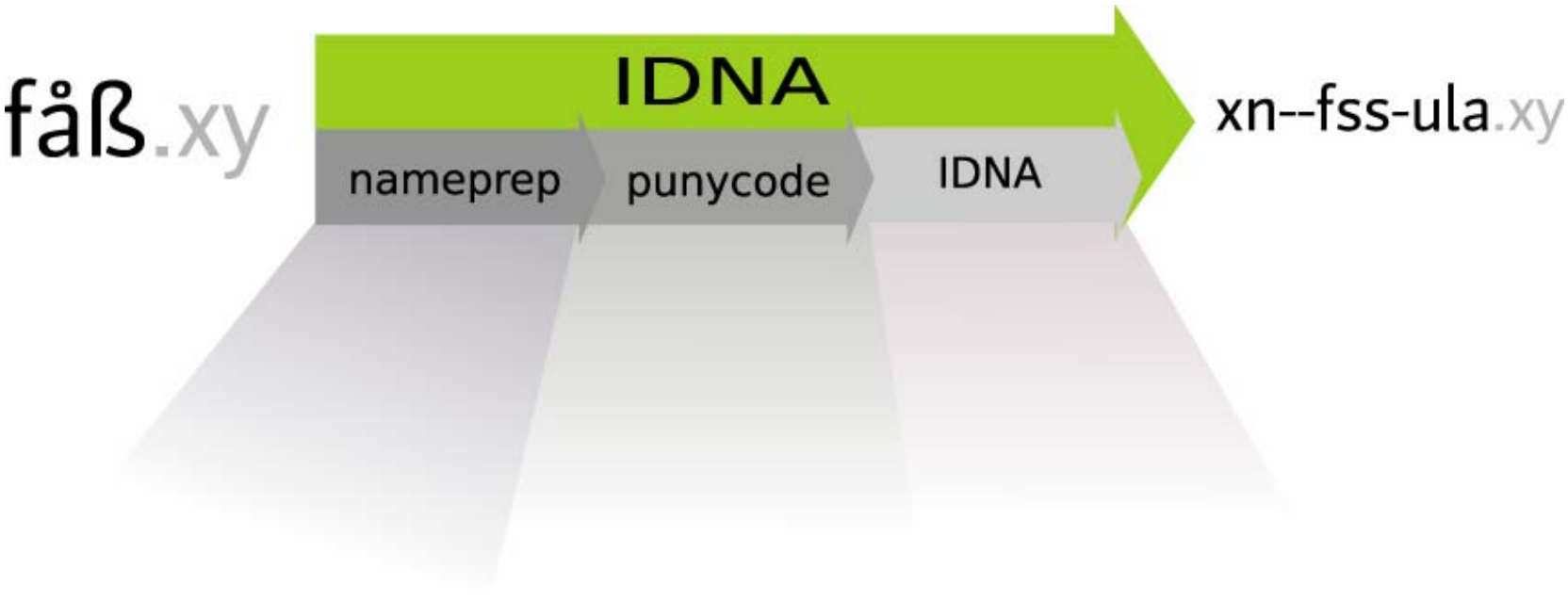


ASCII still transported over the Internet. Unicode never leaves endpoints, travels across Internet as encoded ASCII form.

How does this encoding work

- Takes the expressive Unicode form
- Normalises, compresses, and translates it
- Documented by 3 RFCs – 3490-3492.
  - IDNA
  - *nameprep*
    (A subset of a fourth RFC, *stringprep*)
  - *punycode*
- Why should anyone care?
  - The way these protocols work has an impact on policy choices.

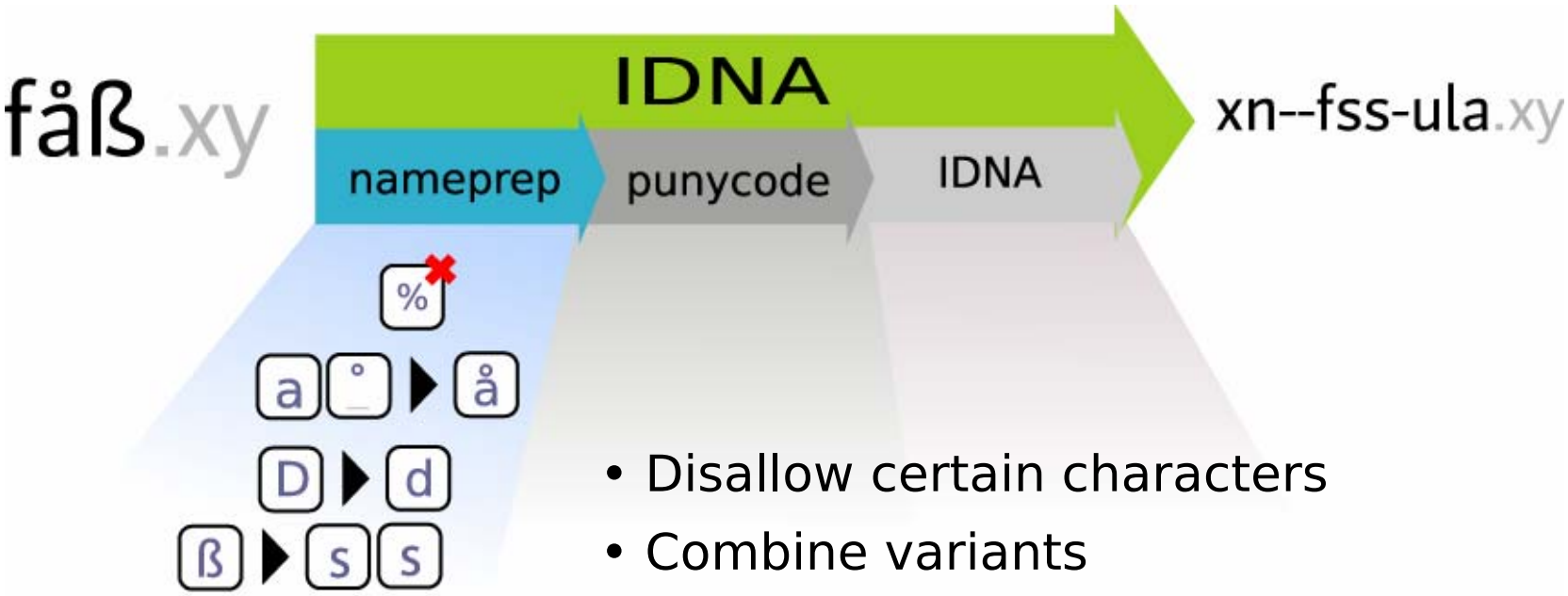
The parts of IDN



- Two essential steps – nameprep and punycode
- Governed by the IDNA specification

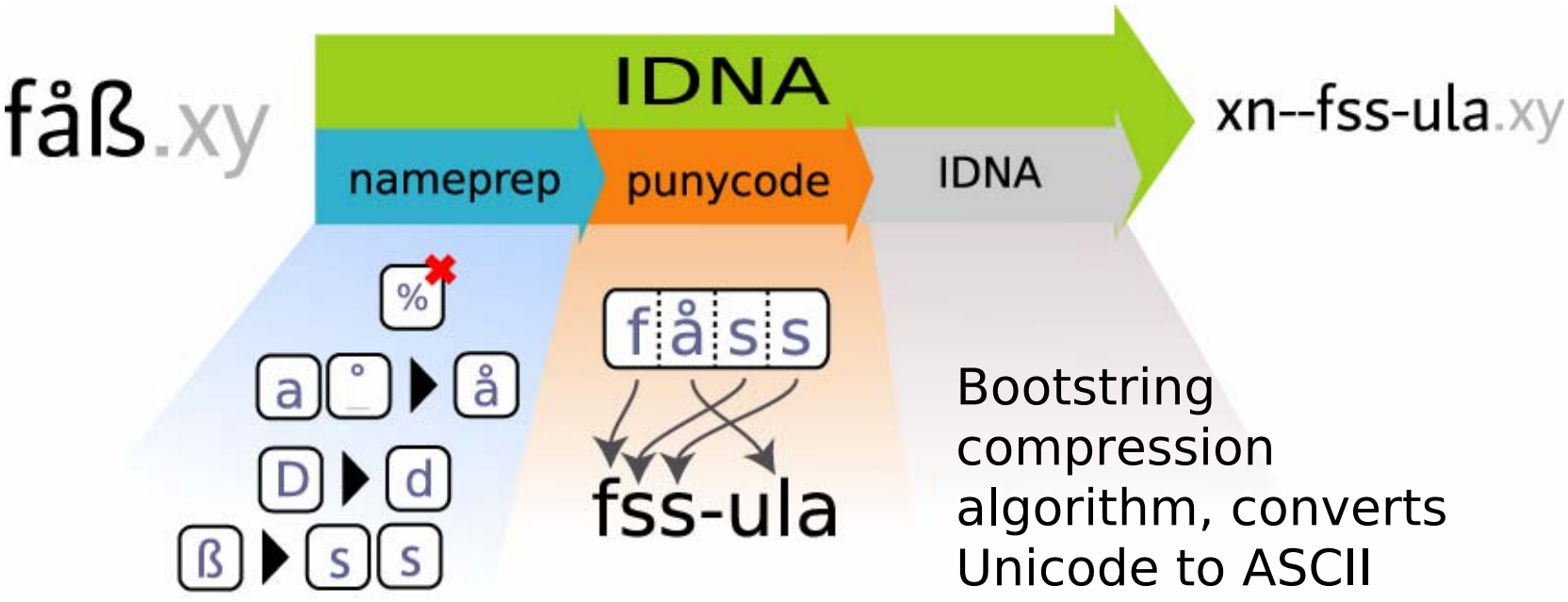
RFC 3491

nameprep



- Disallow certain characters
- Combine variants
- Case folding
- Subset of “stringprep”, RFC 3454

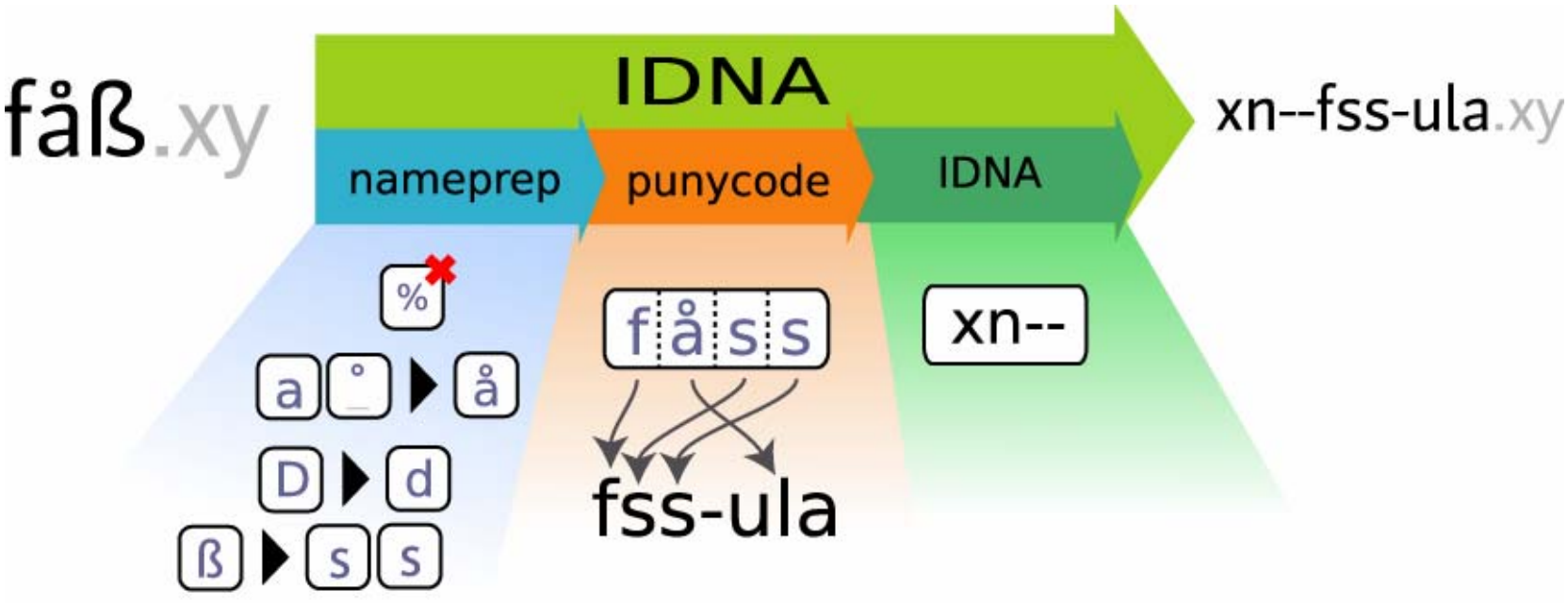
RFC 3492
punycode



Bootstring compression algorithm, converts Unicode to ASCII encoding

RFC 3490

IDNA process



Specifies overall process, iterating over each domain element, prefixing those converted to punycode with “xn--”

The problems with code points

- Unicode is designed to represent every languages
  - Allows you to express most languages with one character set, rather than language-specific
- Varying accuracy at representing specific languages
  - The specifications involves some tradeoffs
  - Different characters are used differently in different languages

Using Unicode as IDN’s basis

- IDNs allow all languages by default
- This may not be desirable
  - Policy reasons
  - Administrative reasons
  - Confusion reasons
- Registries seek to limit the number of allowable code points to those that reflect languages they want to allow registrations for.

How to restrict?

- Identify specific characters required for a language and permit only those.
- Identify a group of characters that represent the same type of letter form (Latin, Cyrillic, Greek, etc.) and permit those. Often referred to as “Code Pages”

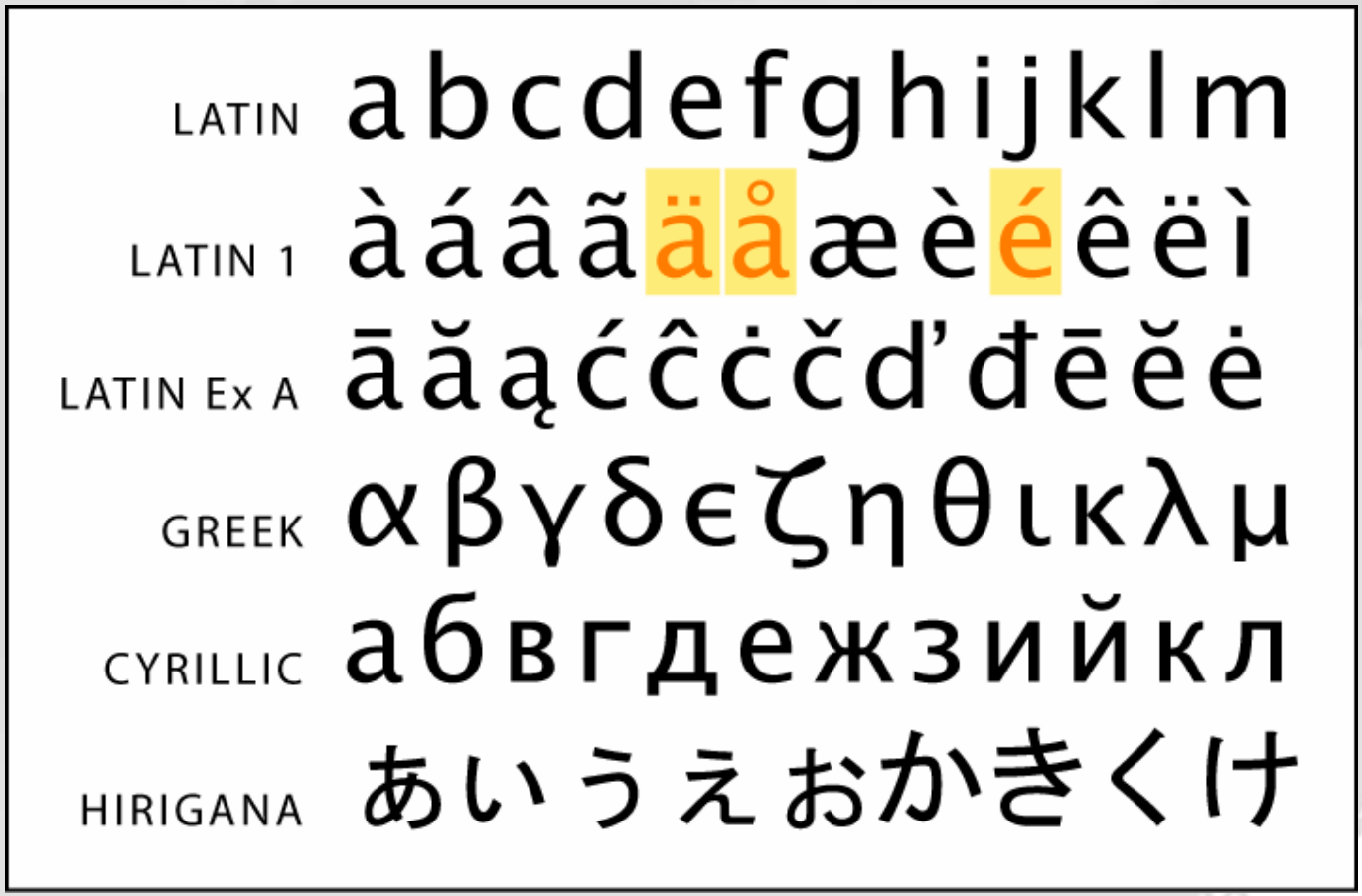
- The less restrictive the set, the more potential for confusing overlaps and “nonsense” being introduced into names.
  - Maybe that is none of the registry’s business?

Restriction registrations

By default, anything goes

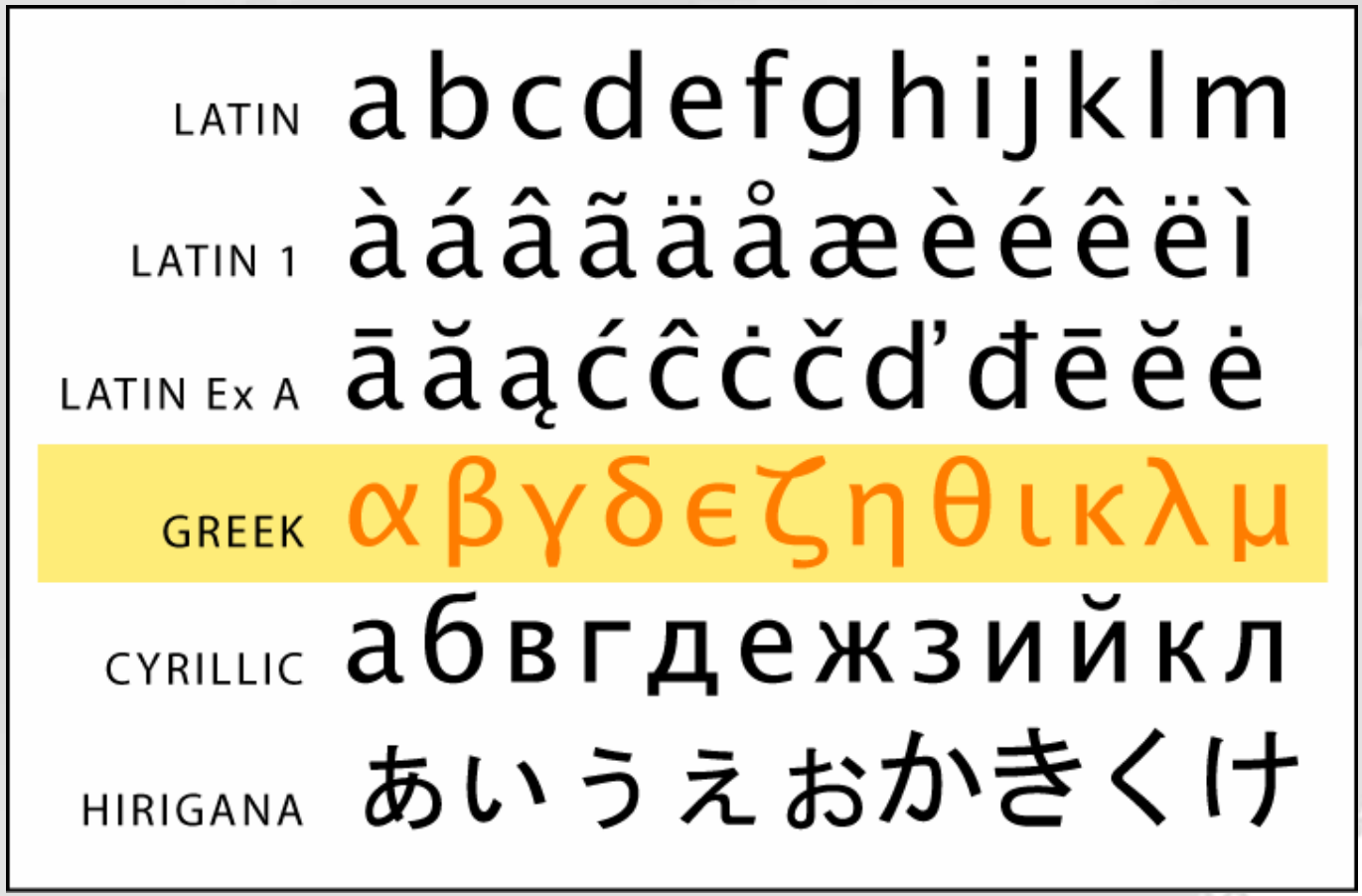
Restricting registrations

One approach: Specific Code Points



Restricting registrations

Another approach: Specific Code Page



One approach:
+92 Latin Code Points

Selection should support most/all Latin-based European languages.

Deployed by German registries (.de, .at, .ch, .li), with some others considering. Poland has this plus more.

Bundles + Variant Tables

- Problem: Confusing alternatives, or equivalent characters, **not** already treated in nameprep process.

- Create a list of code-point alternatives that should not overlap (‘*variant table*’)
- Upon registration, allocate groups of domains using those tables (‘*bundles*’)

Defining the way language works

Øresund

Öresund

ø ↔ ö

Using Bundling

- Registries do something with a registered bundle:
  - Prohibit? No-one is allowed the alternatives.
  - Assign? Associated registant automatically gets the collection.
  - Reserve? Associated registrant can get them separately.
- Seems to be relatively optional in Western Europe. More important in Asian scripts. Greek? Others with optional accents and identical meaning code points.
- Counter arguments:
  - colour.xy vs color.xy;
  - lab.xy vs Iab.xy [lab.xy vs iab.xy]

Another use of bundles

- Iceland to implement bundles as a launch method only. Bundles used to calculate superior rights.

þ → th	á → a
æ → ae	é → e
ð → d	ý → y
ö → o	ú → u
ó → o	í → I

  - eg: gu**d**mundsson.is > gu**ð**mundsson.is

Launching IDNs

Registry Launch Strategy

- Firstly, block IDNs (xn-- prefix) until registry is IDN aware.
- Launching IDNs is like liberalisation
  - More domains become available
  - Likely to be a high demand from people accessing the new domains
  - Under policy or community wishes, some may have preemptive or superior rights to domains
- Launch Event
  - FCFS?
  - Sunrise Periods
  - Lotteries, Auctions, etc

Modifications to registry

- Needs to become Unicode aware
- Special processing for IDNs
  - Check IDNs are valid, optionally check policy compliance.
- Unicode not just in domain field! Email contacts, name server names, etc. May also internationalise address fields at the same time although this is optional.
- Registry interface
  - EPP supports Unicode
  - WHOIS does NOT support Unicode in a standard way
  - Web supports Unicode, but user experience mixed and may not be reliable or predictable.

Policy Issues

- What is the subject of the contractual relationship for the registrant?
  - Unicode form?
  - ASCII form?
  - Both?
  - The bundle of alternatives?
- Maximum/minimum length limits may be redefined
  - 63 octet limit for ASCII form, not Unicode
  - Traditional 3 character minimum does not apply
    - Allowing **á.xy** but not **a.xy** hard to explain

The big problem – supply side

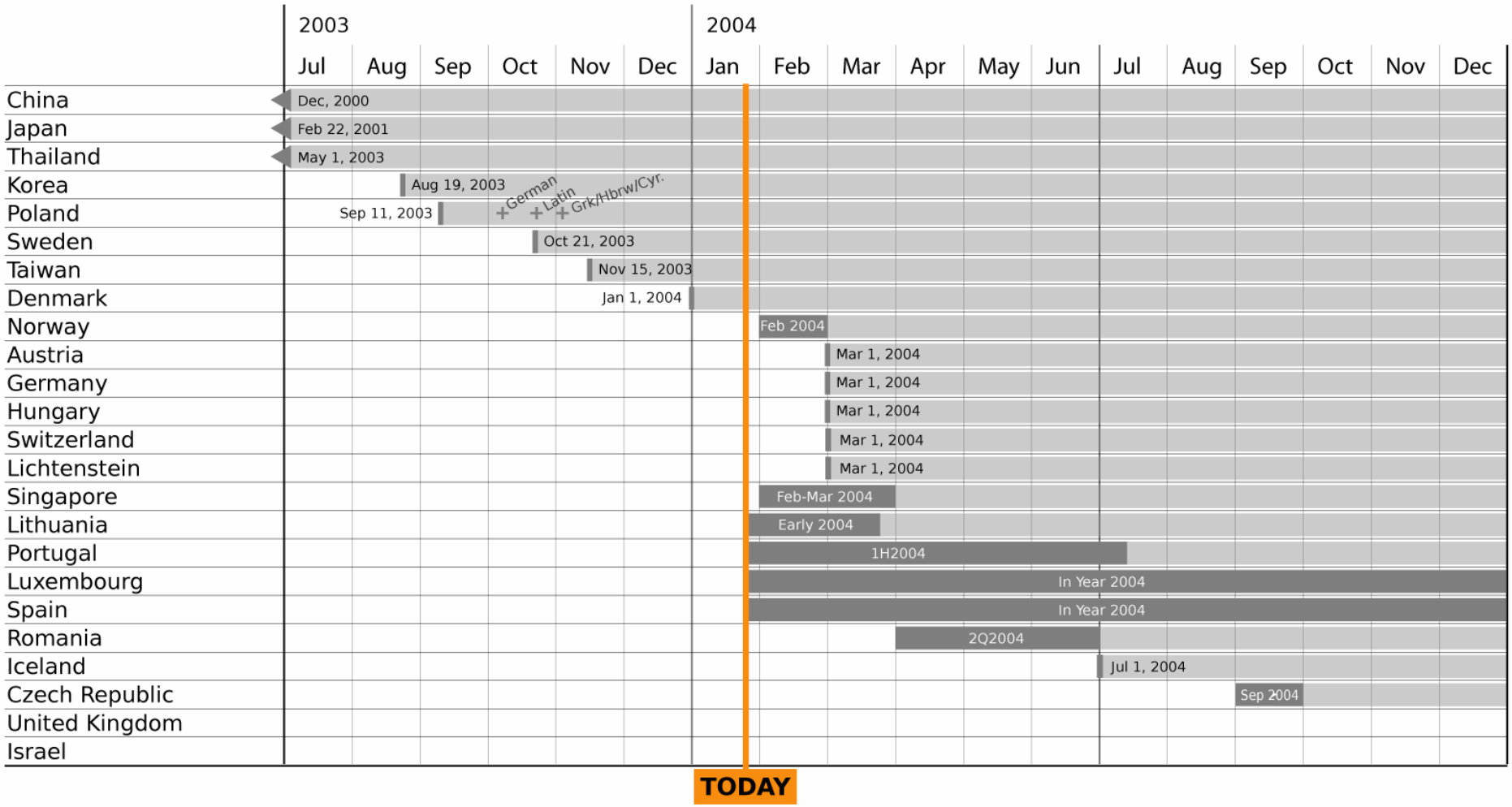
- Many “user interface” issues will head to registrars, end users = relatively uneducated.
  - End user just wants fåß.xy, but needs to know they are getting **xn-fss-ula.xy**
  - Knowledge of IDNs seems low in ISP/services industry. If an IDN is registered, end user must be savvy to use it.
  - Dealing with foreign characters huge customer service challenge.
    - Internationalising business processes, applications
    - Educating customer service staff

The big problem – demand side

- Software support is low
  - Little incentive to use IDNs, when it is a hassle for people to use it
    - Registrations seemed to be technologists and preventative
- “IDN Software Consortium”
  - Seeks to identify ways to spur adoption by software vendors
  - First meeting in Minneapolis
    - Native IE support is key to adoption
    - Microsoft position: Internationalised Resource Identifiers (IRIs) required to make IDNs RFC-legal for use in web addresses.
      - See Dürst draft http://www.ietf.org/internet-drafts/draft-duerst-iri-05.txt
  - www.idnsdc.com

Launch Progress

Where are we now?



Which approaches are being taken?

- Majority (67%) implement code points for local languages.
  - Some choosing specific code pages (Polish, German-speaking..)
- 63% to not implement bundles.

Work within CENTR

- GA2003/20-08-A (“Codepoint” doc)
  - IDN Codepoints for European Languages
  - Still under development
  - Reference document to help develop policy
  - Ultimately subjective, can’t get official decrees for every language
  - http://www.centr.org/meetings/ga-20/idn-codepoints.pdf *(old draft)*

Codepoint Matrix

à	Latin letter A with grave	U+00E0	Catalan French Bokmål Nynorsk Portugese *Norway* *Spain* *United Kingdom*	U+00C0
ą	Latin letter A with ogonek	U+0105	Lithuanian Polish	U+0104
å	Latin letter A with ring above	U+00E5	Danish Bokmål Nynorsk Southern Sami Northern Sami Lule Sami Swedish *Norway*	U+00C5
ã	Latin letter A with tilde	U+00E3	Portugese	U+00C3

Work within CENTR

- GA2003/20-08-D (“Dummies guide” to IDNs)
  - Discusses the issues registries should consider
  - Provide starting points and options
  - Informational, not prescriptive

Some challenges

- WHOIS silent on character set
  - Traditionally US-ASCII assumed, but different registry use different character sets
    - .jp = JIS
    - Some Europeans = Latin-1
  - CRISP is coming
    - However, WHOIS will remain in use for long time to come
- Changing to UTF-8 within databases etc.
- Invoicing software needs updating
- User Interface issues
  - Web display – Unicode conscious systems may not have the fonts to represent properly
  - DENIC found Java UI text-box widget (used in registry system) converts really badly; bugs in JDBC drivers found,
    - wisła.xy → wisla.xy

Loose standardisation of WHOIS approach

- Waiting for CRISP
- In interim
  - Try to come up with a relatively common approach to WHOIS output
  - Most using UTF-8 or Latin-1 (subset)
  - Use DENIC approach?
    - **Ask in ACE, get ACE back**
      $ whois -h whois.denic.de xn--wisa-21a.de
      domain: xn--wisa-21a.de
    - **Ask in UTF8, get UTF8 back (and ACE too?)**
      $ whois –h whois.denic.de wisła.de
      domain: wisła.de
      domain: xn--wisa-21a.de
    - **Allow charset specification with –C flag**
      $ whois –h whois.denic.de –Cutf7 wis+AUI-a.de
      domain: wis+AUI-a.de

Conclusions

What is done?

- IDNs successfully deployed in various registries
- Know most of the registry pitfalls, lots of information sharing

What is not done?

- Lack of IDN support in software
  - Lack of incentive to uptake
- Lack of community awareness of how IDNs work
  - IDN registrations limited to speculative reservations, knowledgeable insiders
  - No real data on impact of IDNs on general community

Questions?