



Internationalizing Internet Identifiers

Martin J. Dürst
MultiMedia Laboratory
Dept. of Computer Science
University of Zurich
Switzerland
mduerst@ifi.unizh.ch
<http://www.ifi.unizh.ch/staff/mduerst/>



Overview

- ◆ Motivations and limitations
- ◆ Syntax legacy
- ◆ Solution alternatives
- ◆ General architecture
- ◆ Solution details
- ◆ Conclusion



Internet Identifiers

- ◆ Domain- and host names
- ◆ Email addresses
- ◆ URIs (URLs, URNs)
- ◆ ...
- ◆ Officially limited to (subset of) ASCII
- ◆ Various ad-hoc uses beyond ASCII
- ◆ Coherent solution needed soon



Motivations for Internationalization

- ◆ Native script is easier to:
 - ◆ Devise (what is my identifier for X?)
 - ◆ Memorize (what was the identifier of X?)
 - ◆ Guess (what could be the identifier of X?)
 - ◆ Understand (what does X refer to?)
 - ◆ Correct (spelling errors)
 - ◆ Identify with (nice identifier, isn't it?)
 - ◆ Manipulate (write, type,...)
- ◆ Because of:
 - ◆ Higher familiarity
 - ◆ No need for transcription



Limitations of Internationalization

- ◆ Limitations should be defined by users, not by technology
- ◆ Unsuccessful identifiers die out quickly
- ◆ Only necessary for *entry points*
- ◆ ASCII only for complete accessibility
- ◆ ASCII only as fallback



Pseudo-Limitations

- ◆ Identifiers should not be meaningful
 - ◆ Design for meaningfulness follows human nature
- ◆ Typability
 - ◆ Relative effort matters
- ◆ Transferability (e.g. e-mail)
 - ◆ Well established for language communities
- ◆ Solution on higher level (directory,...)
 - ◆ Not deterministic, several steps
 - ◆ Will need URL itself



URL Syntax Legacy

- ◆ RFC 1630
 - ◆ Defines URLs as representing characters
 - ◆ Makes examples using ISO 8859-1
 - ◆ Seems to use characters as synonym for octets

- ◆ RFC 1738: three levels
 - ◆ Original characters (arbitrary)
 - ◆ Encoded as octets (arbitrary encoding)
 - ◆ Encoded as characters (very small set + %HH)



URL Syntax Legacy

- ◆ Because protocols use ASCII:
 - ◆ “Coincidental” identity between characters used in the protocol and characters in the URL (widely [ab]used!)
 - ◆ No established encoding convention or correspondence for any characters beyond ASCII

- ◆ URLs are not always in ASCII,
but always characters
 - ◆ Used on servers with EBCDIC file systems
 - ◆ Transmitted in documents using EBCDIC
 - ◆ Accessed from clients using EBCDIC
 - ◆ Written on paper (newspaper, cardboard box, napkin)



Solution Alternatives

- ◆ Scope
 - ◆ General solution
 - ◆ Solution by part/scheme
- ◆ Protocol compatibility
 - ◆ New schemes/protocols (e.g. iftp, imailto,...)
 - ◆ Extension of bit range (7 bits -> 8 bits)
 - ◆ Escaping (e.g. MIME headers)



Identification of “charset”

- ◆ Unspecified (may work locally)
- ◆ Server-specified (e.g. ftp://ftp.server/char.enc returns serverwide encoding)
- ◆ Tagged (e.g. MIME RFC 1522: Dürst -> =?ISO-8859-1?Q?D=FCrst?=)
- ◆ Uniform (e.g. UTF-7, UTF-8)
- ◆ Because URIs are transported on paper, only uniform solution works!



UTF-8 vs. UTF-7

- ◆ Space considerations
(UTF-8 long when escaped)
- ◆ UTF-8 can use URI escaping (%HH),
UTF-7 cannot
- ◆ Many protocols move towards UTF-8,
almost none towards UTF-7
- ◆ UTF-8 officially strongly recommended
by IETF



General Architecture (1)

- ◆ UTF-8 as pivotal representation for URIs
 - ◆ Used for conversion to ASCII fallback/octet
representation (%HH-escaping)
 - ◆ Used for document-independent storage
(and transport)
 - ◆ Used in internal representation (between URI
capture and URI resolution)



General Architecture (2)

- ◆ I18N URI represented as characters independent of UTF-8
 - ◆ On paper
 - ◆ In documents with native encoding and charset labeling (e-mail, HTML,...)
 - ◆ In local UI components



General Architecture (3)

- ◆ For specific protocols and mechanisms
 - ◆ UTF-8 for convergence in “chaotic” protocols
 - ◆ FTP: Officially ASCII only, chaos in practice, new: UTF-8
 - ◆ HTTP: Officially octets, hopefully UTF-8 soon
 - ◆ Could be something else than UTF-8
 - ◆ Adequate single encoding for particular cases
 - ◆ IMAP folder names: modified UTF-7
 - ◆ Domain names: “UTF-5” feasibility proposal
 - ◆ Email addresses: “UTF-5”+escape hatch???
 - ◆ UTF-8 from the beginning for new protocols (ACAP, LDAP)



A Trip of a Japanese URL

- ◆ Document name created in Japanese
(file system uses EUC) and URL constructed
- ◆ Copypasted and sent by email to friend (in JIS)
- ◆ Passed on paper to another person
- ◆ Typed in as HREF attribute
(document encoded in SJIS)
- ◆ Served and converted to ASCII by
transcoding proxy (survives as &#ddd;)
- ◆ Converted to UTF-8 and sent to server
- ◆ Converted by server to EUC and found!



Solution Details (1)

- ◆ Guidelines and specs for “dangerous” and
equivalent characters
 - ◆ Technical equivalences not visible
to humans have to be normalized
(e.g. A-ring vs. A + ring-above)
 - ◆ Backwards compatibility and other unsuited
characters have to be warned against
 - ◆ Rest has to be left to human decisions
(it works for I/l/1; 0/O!)



Solution Details (2)

- ◆ Specifications for BIDI
 - ◆ URI components have to stay in logical order for processing
 - ◆ URIs have to have defined visual order for transferability on paper
 - ◆ Many separators are neutral; context may vary
 - ◆ Define standard “bidi markup” for URIs
 - ◆ LTR overall context
 - ◆ Separators strong LTR by surrounding with LRM
 - ◆ Remove “bidi markup” before resolution



Solution Details (3)

- ◆ Backwards compatibility
 - ◆ Server-side:
 - ◆ Accepts both UTF-8 and legacy encoding
 - ◆ Heuristics for identifying UTF-8 extremely precise
 - ◆ Client-side:
 - ◆ Submit both as UTF-8 and as legacy encoding
 - ◆ Two round trips for entry points and main pages
 - ◆ Reduction possible with heuristics



Solution Details (4)

- ◆ Query parts
 - ◆ Transmitted in body
 - ◆ Philosophically correct
 - ◆ Can be labeled
 - ◆ Solution is being completed
 - ◆ Transmitted as URL
 - ◆ Can serve as entry point
 - ◆ Heuristics difficult because of dense name space
 - ◆ Need temporary labeling (QUERY-UTF-8 in HTTP)



Conclusions

- ◆ Architecture follows ASCII/EBCDIC model with the same distinction between characters and octets
- ◆ Critical mass for UTF-8 is reached
- ◆ Support for UTF-8 is increasing
- ◆ Much work still necessary
 - ◆ Specification (IETF/W3C/UTC)
 - ◆ Implementation