Universal Acceptance of email addresses, domains: how does your framework rate?

Jim DeLaHunt / IUC41 / 18 October 2017

//*.*/

Universal Acceptance

Universal Acceptance

+1,000,000,000

The next one billion internet users use a wide variety of languages and scripts. They will demand email addresses, and domain names, in scripts they can easily read. This is **Universal Acceptance (UA)** — of all domain names and email addresses.

To: مانيش@أشوكا.لهند

To: données@fußballplatz.technology

http:// 普遍接受 - 测试。世界

How does your framework rate?

App development frameworks, libraries, and programming languages on all platforms will be called on to meet the Universal Acceptance challenge. We present technical compliance criteria, ways to evaluate compliance, and a common report format.

☛ Does your library and platform provide Universal Acceptance?

Agenda

Slides: http://go.jdlh.com/iuc41s10t2

* Who we are: UASG, Jim DeLaHunt
* Context: the next one billion, and universal acceptance
* Internationalized Domain Name (IDN) acceptance issues
* Email Address Internationalization (EAI) issues
* Compliance criteria in UASG0018 note
* Next steps
* Q&A

Who we are

Who we are

Universal Acceptance Steering Group (UASG)

* http://www.uasg.tech
* Community-led initiative, world-wide
* Raise awareness, identify problems, solve them
* Project of ICANN, the domain name system organisation

Jim DeLaHunt

* http://jdlh.com, ☎ +1-604-376-8953
* Vancouver, Canada
* Consultant in multilingual websites; software engineer
* UASG volunteer participant

UASG materials available

UASG operates primarily by public education. Participants write outreach materials, technical notes. They give presentations to industry meetings. They evaluate, report, and follow up on UA issue reports.

Technical Notes (selection)

* UASG004 Use Cases for UA Readiness Evaluation
* UASG010 Quick Guide to Linkification
* UASG018 Programming Languages Evaluation Criteria

Plus C-level outreach papers, magazine articles, presentations, ...

Who you are

This talk is primarily for application developers who use frameworks, and for those who develop the frameworks. We assume familiarity with internationalised domain names (IDNA) and email addresses (EAI).

Primary audience

* Application developers, users of domain name and email frameworks
* Framework developers
    * and QA, management, product management, of the above
* Assumes familiarity with IDNA and EAI, but some review

Context

The next 1,000,000,000 Internet users

Next 1 billion

VS

China, India, Third World.
Large share use non-Latin script.
Little marginal North American,
European increase.
Mostly mobile and small-screen,
lower share on desktop, laptop.
Extending to mid-, lower-educated,
less comfortable with Latin script.

First 1 billion

First world, N. America, Europe.
Large share use Latin script.
Includes large share of North
American, European potential.
Mostly desktop & laptop
computers, mobile only later.
Cream of highly-educated in each
market, the best at Latin script

Domain names

Domain names are the primary way to locate things on the internet. Original standards limited domain names an ASCII subset, and thus to Latin script. This obstructs users of non-Latin languages. They aren't just technical (see: ads), or written (see: saying a domain name)

Domain name standards

* ASCII Letters, Digits, and Hyphen, max 63 (RFC1035)
* Well known Top-Level Domains: .com, .org. .net, .jp, .ru, .cn, .in, ...
* e.g. Amazon.com, XgenPlus.com,
* Appear in many areas, e.g. email addresses, URLs, billboards, speech

Domain names, extended

Recent changes permit Internationalized Domain Names for Apps (IDNA). This allows new non-Latin TLDs, and non-Latin characters in rest of name. Parallel changes permit Latin TLDs with more than three characters. Thousands have been registered.

Domain name extensions

* Internationalized Domain Names for Apps "IDNA2008" (RFC5890)
    * Replaces earlier IDNA2003
    * e.g. http:// 普遍接受 - 测试。世界
* .भारत ("bharat", India), . 中国 (China), 「。」 as well as '.'
* .tech, .museum, and hundreds more

Email addresses

Still a mainstay of Internet communication. Actually a stack of related specifications, including SMTP, POP3, IMAP, *etc*. Original standards limited email addresses to an ASCII subset, and thus to Latin script. This obstructs users with names from non-Latin-script languages.

Email standards

* Subset of ASCII, typically letters, digits, punctuation (RFC2822)
* *mailbox* @ *domain.name*, e.g. info@unicode.org
* *mailbox* preferably similar to user's own name in own script
* Many implementations, some deviating from standards

Email addresses, extended

Domain name extensions brings change to the domain.name part of email addresses. Extensions to email address syntax permit almost any Unicode character in mailbox. Consequences ripple through SMTP, MIME, IMAP, POP3, and more.

Email Address Internationalization (EAI) standards

* EAI Overview and Framework (RFC6530) + 6 more RFCs
* EAI requires changes to several protocols and components
* Change takes time, so EAI must interoperate with legacy email

Frameworks matter in app development

Frameworks matter, because in modern app development, most apps will call a framework for domain name and email address handling rather than writing that code directly. Limits in the framework become limits of your application.

* Framework == library == module, terminology differs
* Your application also handles email addresses and domain names
    * Especially for input from and presentation to the user
* Stating the obvious here

Internationalized Domain Name (IDNA) acceptance issues

IDNA Handling: your app + your framework

To provide Universal Acceptance, your framework must handle the low level domain name operations correctly, and your application must handle the high level operations correctly. Here is a quick guide to the high level operations.

http://普遍接受-测试。世界

Learn the IDNA reference knowledge

Learn about Internationalized Domain Names for Applications, understand Nameprep and Punycode, know when to use U-Labels and A-Labels. In due course, libraries should take over some of this for you.

IDNA references

* RFC5890 IDNA: Definitions and Document Framework
* RFC5891 IDNA: Protocol
* RFC5892 The Unicode Code Points and IDNA
* RFC3492 Punycode: A Bootstring encoding of Unicode for IDNA
    * etc....

Tools & Resources for Developers

Authoritative Tables:

* http://www.internic.net/domain/root.zone
* http://www.dns.icann.org/services/authoritative-dns/index.html
* http://data.iana.org/TLD/tlds-alpha-by-domain.txt
* See SAC070 on static TLD / suffix lists: https://tinyurl.com/sac070

Internationalized Domain Names for Applications:

* Tables: https://tools.ietf.org/html/rfc5892
* Rationale: https://tools.ietf.org/html/rfc5894
* Protocol: https://tools.ietf.org/html/rfc5891

Unicode:

* Security Considerations: http://unicode.org/reports/tr36/
* IDNA Compatibility Processing: http://unicode.org/reports/tr46/

Universal Acceptance Steering Group info & recent developments: www.uasg.tech

//*.*/

Five Key Tasks of Universal Acceptance

Accept. Validate. Store. Process. Display. For all domain names.

Make wise end-to-end decisions about using A-Labels, U-Labels.

UASG guides

* UASG006 Universal Acceptance Quick Guide

http:// 普遍接受 - 测试。世界

Linkification and Universal Acceptance

When you recognise URLs or IRIs and automatically make them links ("linkification"), do so with universal acceptance. If you have a detailed regular expression in your code, it is probably wrong. We have a guide.

http://اختبار-القبولالعالمي.شبكة

UASG guides

* UASG010 Quick Guide to Linkification
  * Standards Principle: link all well-formed URLs
  * Universal Acceptance Principle: treat all top-level domains & all scripts well
  * Safe Practice Principle: various security considerations
    * etc....

Email Address Internationalization (EAI) issues

EAI Handling: your app + your stack

To: مانيش@أشوكا.لهند

To provide Universal Acceptance, your framework must handle Email Addresses Internationalization correctly. The various components which make up your email sending and receiving stack must also be support EAI. Here is a quick guide to the high level issues.

To: données@fußballplatz.technology

//*.*/

EAI Universal Acceptance in your app

Email addresses can have international text in both the mailbox and domain name parts. If app tests email addresses with a detailed regular expression in your code, it is probably wrong. We have a guide.

UASG guides

* UASG014 Quick Guide to Email Address Internationalization (EAI)
  * Clients (Mail User Agent) display domain name in Unicode, send as A-Label; display and send mailbox name in Unicode
  * Follow UASG010 Linkification guide for links in messages
  * Consider validation via test emails rather than by address structure

EAI Universal Acceptance in your stack

Your stack of email sending and receiving components need to support EAI, and declare that they do. They may include SMTP, IMAP, POP3, and more. We have a guide and case studies.

UASG guides

* UASG014 Quick Guide to Email Address Internationalization (EAI)
  * Servers (Mail User Agent) advertise SMTPUTF8 support etc.
  * Email service providers, consider ASCII alternative addresses, proper casing
  * Transition challenges with non-EAI correspondents

EAI case studies

We have case studies of organizations which have already supported EAI. Their experience helps you know what to expect. They may have tools you can use to help test your EAI.

UASG guides (partial)

* UASG013D Case Study: Data Xgen Technologies Pvt Ltd
  * "updating… at least 12 major elements… webmail, IMAP, POP, SMTP, contacts, calendar, antispam, search, logger and rules."
* UASG013C Case Study: ICANN
  * Phased approach, 87 components = 46 in-house + 41 from vendors

Compliance criteria in UASG0018 note

UASG0018 document

//*.*/
Universal Acceptance

Reviewing programming languages and frameworks for compliance with Universal Acceptance good practice

21 August 2017
Version 1.0

Report UASG Programming Language Framework Review

Reviewing programming languages and frameworks for compliance with Universal Acceptance good practice
https://uasg.tech/documents/

Library Evaluation

A library evaluation is a report with 5 sections:

1. Basic information
2. Auxiliary information
3. Implementation notes
4. Technical evaluation
    a) 10 Test suites (based on behaviour described in RFCs)
    b) Report on ‘Semantic Checks’ functionality
5. Mitigation Actions

UASG0018 test suites

Low-level functions

* L-U2A: IDNA2008 - Convert Uni. domain name to ASCII lookup form
* L-R2A: IDNA2008 - Convert registration label to ASCII registry form
* L-A2U: IDNA2008 - Convert ASCII domain name to Unicode
* L-DNC: IDNA2008 - Domain name equivalence comparison

High-level functions

* H-DNS: Domain name - syntactic check
* H-DND: Domain name - decompose into components
* H-ES: Email address - syntactic check
* H-ED: Email address - decompose into components
* H-US: URL - syntactic check
* H-UD: URL (IRI): decompose into components

Example test: L-A2U, javascript

```
'use strict';

var uts46 = require('idna-uts46');
var ascii = "xn----f38am99bqvcd5liy1cxsg.xn--rhqv96g";
var unicode = uts46.toUnicode(ascii);

console.log("DNS " + ascii + ": " + unicode);
```

```
$ js example.js
DNS xn----f38am99bqvcd5liy1cxsg.xn--rhqv96g: 普遍接受-测试.世界
```

UA use cases

IDNA Pattern	Example
ascii.long	ua-test.technology
idn.idn	普遍接受 - 测试 . 世界
idn-rtl.idn-rtl	اختبار- القبولالعالمي. شبكة
idn.ascii/unicode	普遍接受 - 测试 .top/ 我的页面
EAI Pattern	Example
unicode@idn.idn	युएअसजी@डाटामेल.भारत
ascii@ascii.idn	info4@ua-test 。世界
unicode@rtl.rtl	دون@رسيل.السعودية

UASG004 Use Cases
These domains are registered, ready to use in test suite.
Total 45 cases.

Scoring system

Evaluation items are rendered as 0..5 points, then summed.

1. Basic information (not scored)
2. Auxiliary information (scored 0..5, weighted 50%)
3. Implementation notes (scored 1..5, weighted 50%)
4. Technical evaluation
   a) Test suites (10 suites)
      - scored as test pass rate for each suite, 0..100%
      - normalised to 0..5 points per suite

   b) Report on 'Semantic Checks' functionality (not scored)
5. Mitigation Actions (not scored, reported as list of actions to take)

Next steps

Candidates for Evaluation

C language

* GNU libidn2: IDNA2008
* GNU libidn: IDNA2003

Python language

* idna module: IDNA2008
* encodings.idna module
  * Python 3.x, Python 2.7.x

PHP language

* IDN functions:
  * IDNA2008, IDNA2003

Go language

* idna package: IDNA2008

Javascript language

* Idna-uts46 npm package
* Bundled with node.js
* IDNA2008

Plans for UASG0018 work

Do evaluations

* Perform evaluations and publish results
* You could contribute an evaluation!

Collaborate with framework maintainers

* Contribute UA test suites to framework test suite
* Track framework bug reports based on UA test results

Improve UASG0018

* Reconsider high-level vs low-level split, and test nomenclature
* Review scoring system after a few reviews
* Maybe add test environment to library evaluation report

How you can help

Evaluate the frameworks you use! And do so in collaboration with UASG, so that your evaluation can help others, and lead to improvements in the frameworks you use. Join us!

Suggested next steps for you

* Use UASG0018 to write a library evaluation useful to you
* Follow the UASG at https://uasg.tech/ .
    * *e.g.* https://uasg.tech/event/webinar-broccoli-issues/ , 19. Oct, 14h UTC-4
* Subscribe to the *ua-discuss@uasg.tech* email list.
    * https://mm.icann.org/mailman/listinfo/ua-discuss

Q&A

Thank you!

Q&A

Slides: http://go.jdlh.com/iuc41s10t2

Join us! https://uasg.tech/

Evaluation: http://unicodeconference.org/eval-sp/

//*.*/

Universal Acceptance of email addresses, domains: how does your framework rate?

Jim DeLaHunt / IUC41 / 18 October 2017