Draft - Introduction to Universal Acceptance and its Technical Remediation

January 2020

ICANN

Agenda

- Tutorial Target Audience and Objectives
- Key Fundamental Aspects: Unicode, IDNs, EAI, UA
- Email Address Internationalization (EAI)
- Developing Applications to Support UA in Java
- Developing Applications to Support UA: Best Practices
- Summary
- References

Tutorial Target Audience and Objectives

- Target Audience: Technologists
- Objectives:
  - Understand base key concepts related to Universal Acceptance (UA).
  - Understand Email Address Internationalization (EAI).
  - Understand Java programming for UA.

Key Fundamental Aspects: Unicode, IDNs, EAI, UA

Plan

- Key fundamental aspects (just the necessary concepts)
  - Unicode, UTF-8, Normalization
  - Domain names, labels, top-level domains (TLDs), zones
  - Internationalized Domain Names (IDNs): Punycode, U-label, A-label.
  - Universal Acceptance (UA)
  - Email agents: MUA, MTA, etc.

Note

- In this section, there will be short summaries on multiple topics.
- Please keep in mind, each topic could be covered in a full-day tutorial, so these summaries are kept simple and cannot be considered complete.
- These summaries are meant to make sure that key concepts can be understood before the main content on EAI and Java programming.

Unicode

- Encoding glyphs into codepoints.
- In specifications, codepoints are shown in hex using the U+XXXX notation.
- Codepoints are typically carried using the UTF-8 (Unicode Transformation Format, 8 bit) format.
  - Variable number of bytes for a single codepoint.
  - ASCII is used as is.
  - Gold standard for carrying Unicode codepoints in web, protocols, etc.

Unicode

- There are multiple ways to use a glyph:
  - "è" = U+00E8
  - "e`" = "è" = U+0065 U+0300
  - Normalization is a process to ensure that no matter the user type, the end representation will be the same.
    - For the two entries above, Normalization Form C (NFC) will generate U+00E8 for both.

Internationalized Domain Names (IDNs)

- Internationalized Domain Names (IDNs) enable the use of non-ASCII characters for any label of a domain name.
  - Not all labels of a domain name may be internationalized.
- Example: exâmple.ca
- User uses the IDN version, but the IDN is converted into ASCII for DNS resolution.
  - exâmple => exmple-xta => xn--exmple-xta
  - The xn-- prefix is added to identify an IDN.

Internationalized Domain Names (IDNs)

- Example process of using an IDN:
  - User enters in a browser: http://exâmple.ca
  - Browser does normalization on the user entry.
  - Browser converts exâmple.ca in an ASCII compatible representation called Punycode [RFC3492], and adds 'xn--' in front of it.
    - exâmple.ca becomes: xn--exmple-xta.ca
  - Browser calls the DNS to get the IP address of xn--exmple-xta.ca
  - Browser connects to the HTTP server at the received IP address.

Internationalized Domain Names (IDNs)

- The protocol for handling IDNs is named IDN for Applications (IDNA).
  - Two versions: IDNA2003 and IDNA2008. The latter (IDNA2008) is the one currently used.
- U-label is the Unicode native representation of an IDN label: exâmple
- A-label is the Punycode representation of an IDN label: xn--exmple-xta

Two IDN Standards

- IDNA2003 (RFC3490)
  - Was defined against Unicode 3.2 (March 2002).
  - Newer characters defined after Unicode 3.2 are accepted as is.
- IDNA2008
  - Based on Unicode codepoint properties.
    - New characters are automatically handled by their properties.
  - Therefore, fewer number of characters are accepted in IDNA2008 than IDNA2003.

Public Suffix List (PSL)

- The Public Suffix List (PSL) is an attempt to help web developers know whether or not domains are controlled by the same organization. This can be thought of as asking: does this domain name allow others to register under it?
- https://publicsuffix.org/
- Some libraries, frameworks, and applications use the PSL as a static list of TLDs.
- If used in applications:
  - The developer has to keep the copy up-to-date in its application.
  - A TLD may not be listed in the PSL.
- To know if a name is actually a TLD, use other means.

Email Address Internationalization (EAI)

- Email syntax: leftside@domainname
- Domain name can be internationalized as an IDN.
- Left side (also known as local-part or mailbox name) with Unicode (UTF8) is EAI.
- Examples:
  - kévin@example.org
  - すし@快手.游戏

Email Address Internationalization (EAI)

- Side effect:
  - Mail headers need to be updated to support EAI.
  - Mail headers are used by mail software to get more information on how to deliver email.
- Since not every email server supports EAI, a negotiation protocol is used to only send EAI when the target server supports it. If not, then it falls back and returns an unable to deliver message back to the sender.
- The SMTPUTF8 option is used within the mail transfer protocol (SMTP: Simple Mail Transport Protocol) to signal the support of EAI by an email server.

Universal Acceptance (UA)

- UA is about how to appropriately support internationalized identifiers, as well as new and long TLDs.
  - Internationalized identifiers:
    - IDN
    - EAI

Universal Acceptance (UA)

- UA is also about:
  - New and longer string TLDs (new generic top-level domains – new gTLDs)
  - In the early days, TLDs were either two or three characters long (.ca, .com). Recently, TLDs started to have longer strings (.info, .google).
  - A TLD label may have up to 63 octets.
  - Some applications are still verifying that the TLD entered by a user has a maximum of 3 characters.
  - Other ones are using regex which takes a maximum of 6-7 characters for the TLD.

Universal Acceptance (UA)

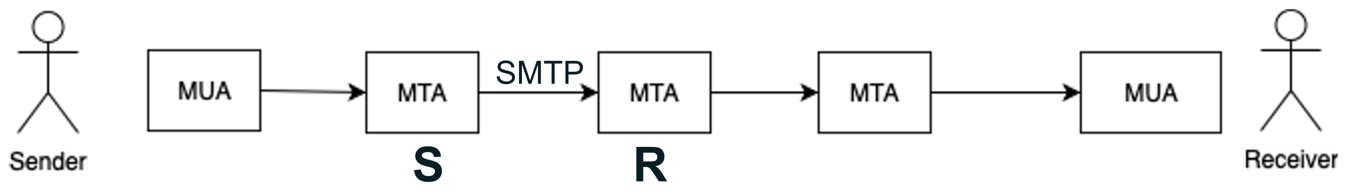
- UA is also about:
  - Added/removed TLDs:
    - TLDs come and go on a « daily » basis.
    - Some applications are verifying the correctness of a TLD based on a static list which is not the latest one, therefore making wrong assumptions about the existence of a TLD.

Email Address Internationalization (EAI)

EAI Protocol Changes

- SMTP
  - Is augmented to support EAI.
  - Has a signaling flag (SMTPUTF8) to specify support of EAI.
  - All SMTP servers in the path must support EAI to successfully deliver the email.
- POP/IMAP
  - Are augmented to properly support EAI.
  - Have a signaling flag to specify support of EAI.

SMTPUTF8 Example



Server S forwarding an email to server R

Specific SMTPUTF8 Signaling (EAI support)

S: <connect>
R: 220 receive.net ESMTP
S: EHLO sender.org
R: 250-8BITMIME
R: 250-**SMTPUTF8**
R: 250 PIPELINING
S: MAIL FROM:<猫王@普遍接受-测试.世界> **SMTPUTF8**
R: 250 Sender accepted
S:RCPT TO:<ray@receive.net>
R:250 Recipient accepted

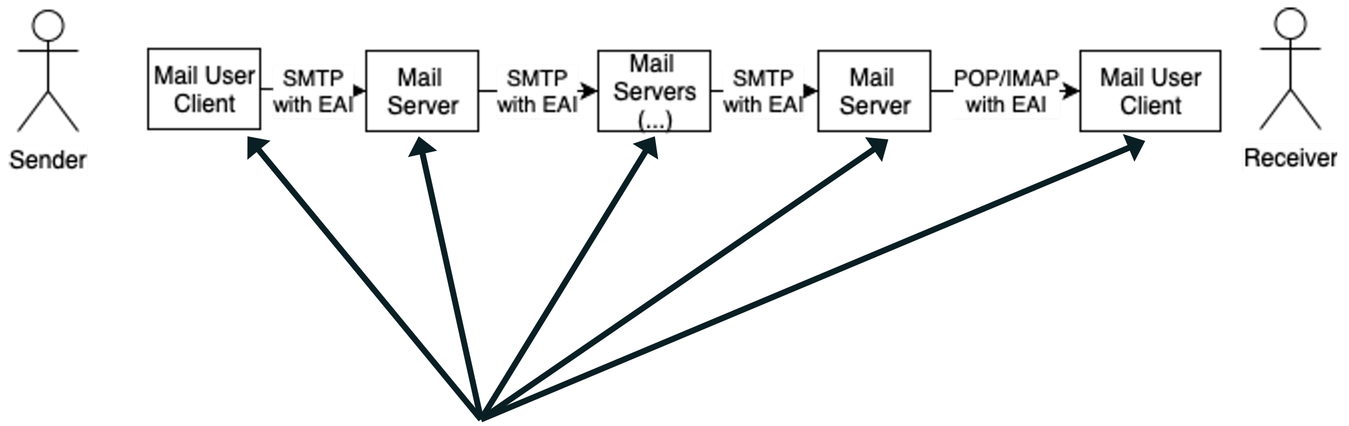
SMTPUTF8 Example

```
S:DATA
R:354 Send your message
S:From: 猫王 <猫王@普遍接受-测试.世界>
S:To: ray@receive.net
S:Subject: 我们要吃午饭吗?
S:
S:How about lunch at 12:30?
S:.
R:250 Message accepted 389dck343fg34
S:QUIT
R:221 Sayonara
```

Email itself

Note: example from UASG012 but corrected.

Protocol Changes, Delivery Path Considerations



To send and receive an email with EAI:

- <u>All email parties involved in the delivery path have to be updated for EAI support</u>.
- If a single SMTP server in the path does not support EAI, then the email is not delivered.

Protocol Changes, Delivery Path Considerations

- What happens when one email (SMTP) server in the path does not support EAI?
  - The last server trying to send to the next hop:
    - Sends back to the sender user a report of unable to deliver.
    - Drops the email.
  - Similar to reports that a sender receives when an email address does not exist.

Considerations

- Case folding:
  - In ASCII, email users expect the equivalence of lowercase and uppercase.
    - Example: PETER@example.com and peter@example.com will be delivered to the same mailbox.
  - Typically for EAI, the case folding functionality is not implemented in most EAI-ready software.
- SPAM:
  - EAI emails may be considered as spam by spam filtering software even when proper SPF/DKIM records are enabled.
- Software/Services:
  - Not every server/client software and services support EAI.

Developing Applications to Support UA

Developing an Application to Support UA

- Key touch points to support UA:
  - Validating user input when it could be an internationalized identifier.
    - For example, emails are often used as the primary identifier to « login » into an application.
  - Processing the internationalized identifiers within the application code path.
  - Storing the internationalized identifiers (in databases).
  - Displaying the internationalized identifiers to the user.

- See UASG007 and UASG026 documents.

Validating User Input

- Traditionally, developers expected:
  - ASCII domain names.
  - ASCII local part of an email address.
- Domain or email user input validation was often implemented by simple regular expressions (regex).
- Strict regex for an IDN and EAI is pretty complicated to achieve.
- An IDNA2008 library is the only right tool to validate IDN domain names.
- Either:
  - Do not use a regex and use an IDNA library to validate the domain.
  - Use a simple regex (with no restriction for local part on email address) and use an IDNA library to validate the domain.

Using DNS Calls

- Traditionally, OS DNS query library calls such as gethostbyname() expect ascii domains.
- These calls can still be used if pre-processing is done:
  - Validate domain name.
  - Convert the U-labels into their A-label version.
  - Pass the resulting domain name to the DNS query library call.

Using High-Level Libraries and Frameworks

- Currently, most OS, especially on the mobile side, provide high-level frameworks for handling domains and URLs.
  - Unfortunately:
    - Not all libraries and frameworks are validating/processing IDN or EAI.
    - In the Java world, many libraries and frameworks support the old IDN standard: IDNA2003.

Some Java Libraries and Frameworks

- JRE-IDN (import java.net.IDN) implements IDNA2003.
- Apache Commons Validator (import org.apache.commons.validator) have a static list of TLDs for validation; therefore, the library is always outdated in an application.
- ICU (import com.ibm.icu.text.IDNA) implements IDNA2008 as a wrapper on the C library, which makes its use in Java well integrated. This is the best library for now.
- Guava (import com.google.common.net.InternetDomainName) uses a copy of the PSL static list; therefore, the library is always outdated as bundled in an application.
- Java URL/URI (import java.net.URL) do not validate hostnames. Use another library for validation.

Developing Applications to Support UA: Best Practices

Best Practices

- Validation for EAI, IDN, UA input is complicated.
- Never rely on a static list of TLDs as they come and go. More to come.
- Do not code any specific syntax or length other than what is in the standards. For example, a label such as a TLD, may have a length of up to 63 octets in A-label/ASCII format which includes the ‘xn--’ prefix for an IDN.
    - Coding a TLD to have a maximum of 6 or 7 octets is just wrong.
- Use string type that is UTF8 to hold domain name and email addresses.
- Make sure to do normalization of input before storing, comparing, and processing.
- If storing identifiers in a database, make sure the whole code path up and including the database itself is UA compliant. For example, UTF8 should be the charset of the column storing the identifier (domain or email) in SQL databases.

Best Practices

- It is preferred to do some basic validation and then make requests and trap the errors, rather than trying to do too much validation.
- Use a UA-conformant library/framework (IDNA2008 for domains).
- Do unit and system testing of UA identifiers.
- Consider using UASG004 as a good starter for test cases.
- Consider converting domain names to their A-label equivalent when passing to libraries, so it is safer to go through the whole code path (that may include libraries that are dependencies and are not UA conformant).

Domain Name Validation and Resolution

- Convert U-labels into A-labels and then pass to the standard methods.
- Think of U-labels as for display.
- There is a one-to-one direct lossless conversion between A-labels and U-labels, therefore no need to keep both. Keep A-labels as they are supported more in the code and dependencies.
- However, U-labels are useful for sorting, comparing, and searching as they will be sorted based on the real value.
    - i.e. the UTF8 string instead of its Punycode encoding.
- When you're about to display a string, always convert to U-labels as the end user expects them. The libraries will do nothing if the domain is in A-label format.

Email Validation and Resolution

- Do normalization of the UTF8 local parts if the email is received as input.
- Always use normalized local parts when comparing, sorting, or searching.
- For the domain part, see domain section.
- Validate using the right lib before sending.
- If sending an email to an EAI address, prepare for a situation where:
  - The email could be refused by your mailer.
  - The email may not reach its destination if one of the mail servers in the path does not support EAI.

Summary

Summary

- UA is:
    - Internationalized identifiers (IDNs, EAI)
    - New and long TLD strings
    - Dynamic list of TLDs
- When dealing with UTF8 strings, always normalize before processing, compare, sort, store.
- IDN labels have two representations: A-label and U-label. U-Label is used for display and compare/sort/search; A-label for processing.
- IDNA has two version: IDNA2003 and IDNA2008. Always use IDNA2008.
- EAI is local part in UTF8.
- Do not use code/libraries that have a static list of TLDs as these change often.
- Do not use regex for user input validation of internationalized identifiers. Use IDNA2008 libraries for IDN; EAI local part may be difficult to validate.

References

UA References

- Use Cases for UA Readiness Evaluation, UASG004
- http://uasg.tech/
- Reviewing Programming Languages and Frameworks for Compliance with Universal Acceptance Good Practice, UASG018
- Evaluation of Software libraries for UA Readiness: http://uasg.tech/software

IDN References

- Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.
- Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://www.rfc-editor.org/info/rfc5891>.
- Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.17487/RFC5892, August 2010, <https://www.rfc-editor.org/info/rfc5892>.
- Alvestrand, H., Ed., and C. Karp, "Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)", RFC 5893, DOI 10.17487/RFC5893, August 2010, <https://www.rfc-editor.org/info/rfc5893>.
- Klensin, J., "Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale", RFC 5894, DOI 10.17487/RFC5894, August 2010, <https://www.rfc-editor.org/info/rfc5894>.
- Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, DOI 10.17487/RFC3492, March 2003, <https://www.rfc-editor.org/info/rfc3492>.

EAI References

- Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 6530, DOI 10.17487/RFC6530, February 2012, <https://www.rfc-editor.org/info/rfc6530>.
- Yao, J. and W. Mao, "SMTP Extension for Internationalized Email", RFC 6531, DOI 10.17487/RFC6531, February 2012, <https://www.rfc-editor.org/info/rfc6531>.
- Yang, A., Steele, S., and N. Freed, "Internationalized Email Headers", RFC 6532, DOI 10.17487/RFC6532, February 2012, <https://www.rfc-editor.org/info/rfc6532>.
- Hansen, T., Ed., Newman, C., and A. Melnikov, "Internationalized Delivery Status and Disposition Notifications", RFC 6533, DOI 10.17487/RFC6533, February 2012, <https://www.rfc-editor.org/info/rfc6533>.
- Levine, J. and R. Gellens, "Mailing Lists and Non-ASCII Addresses", RFC 6783, DOI 10.17487/RFC6783, November 2012, <https://www.rfc-editor.org/info/rfc6783>.
- Resnick, P., Ed., Newman, C., Ed., and S. Shen, Ed., "IMAP Support for UTF-8", RFC 6855, DOI 10.17487/RFC6855, March 2013, <https://www.rfc-editor.org/info/rfc6855>.
- Gellens, R., Newman, C., Yao, J., and K. Fujiwara, "Post Office Protocol Version 3 (POP3) Support for UTF-8", RFC 6856, DOI 10.17487/RFC6856, March 2013, <https://www.rfc-editor.org/info/rfc6856>.
- Fujiwara, K., "Post-Delivery Message Downgrading for Internationalized Email Messages", RFC 6857, DOI 10.17487/RFC6857, March 2013, <https://www.rfc-editor.org/info/rfc6857>.
- Gulbrandsen, A., "Simplified POP and IMAP Downgrading for Internationalized Email", RFC 6858, DOI 10.17487/RFC6858, March 2013, <https://www.rfc-editor.org/info/rfc6858>.

Engage with ICANN

Thank You and Questions

Visit us at **icann.org**
Email: email

@icann	linkedin/company/icann
facebook.com/icannorg	slideshare/icannpresentations
youtube.com/icannnews	soundcloud/icann
flickr.com/icann	instagram.com/icannorg