Introduction to Universal Acceptance

Universal Acceptance is the ability to Accept, Store, Process and Display all Top Level Domains equally and all IDNs, IRIs and email addresses equally.

Contents

Baseline Concepts

Domain Name System (DNS)

Each resource on the Internet is assigned an address to be used by the Internet Protocol (IP). Since IP addresses are difficult to remember, servers collectively providing a public Domain Name System (DNS) exist at well-known addresses on the Internet. DNS provides the mapping between IP addresses and human-readable domain names.

Top level Domains (TLDs)

Human readable domain names are managed by companies known as registrars. When one registers a domain, it will consist of multiple text strings representing multiple domain levels, each separated by a "." character. In most scripts, the right-most domain level is the top-level domain (TLD).

Examples of well-known TLDs

- `.com`
- `.gov`
- `.info`
- `.org`

Some TLDs are assigned to specific countries, and are called Country Code TLDs.

Examples of ccTLDs

- China = `.cn`
- Russia = `.ru`
- United States = `.us`

Generic Top Level Domains (gTLDs)

Starting in 2013, the organization responsible for the creation and maintenance of TLD assignments has generated a large number of new TLDs. Some of these new TLDs are for brands, and others are generic. Nevertheless, all of these new TLDs are usually known as Generic Top Level Domains (gTLDs).

Examples of new gTLDs

- `.apps`
- `.lawyer`
- `.shopping`
- `.panasonic`
- `.osaka`

ASCII & Unicode

In the examples above, all of the text strings are represented using the English character set. This character set is included in American Standard Code for Information Interchange (ASCII, or US-ASCII) character-encoding scheme. ASCII is an older encoding scheme and was based on the English language. For historical reasons it became the standard character encoding scheme on the Internet. ASCII uses only 7 bits per character, which limits the set to 128 characters.

Because most languages do not use the English character set, alternate encodings have subsequently been adopted. The Universal Coded Character Set (Unicode) is capable of encoding more than 1M items. Each of these Unicode items is a called a code point. The most common encoding is called Universal Coded Character Set Transform Format 8-bit (UTF-8).

Character Sets and Scripts

TBD

Internationalized Domain Names (IDNs) & Punycode

By using Unicode instead of ASCII, domain names can contain non-English characters. Domain names using any non-ASCII characters are called Internationalized Domain Names (IDN). Unfortunately, DNS itself still uses ASCII, so an additional encoding was created to allow Unicode code points to be transformed back into ASCII. The resulting strings can be quite large due to the number of valid Unicode code points. The algorithm which implements this Unicode-to-ASCII encoding is called Punycode, and domain names thus encoded can be distinguished from ordinary ASCII because they always start with the characters "`xn--`". (The Punycode transformation is bi-directional. It can transform from Unicode to ASCII and from the xn - - ASCII back to Unicode)

The internationalized portion of an IDN can be in any level, not just the top level. Some gTLDs are IDNs, but not all IDNs are TLDs.

Examples of (imaginary) IDNs

- everyone.みんな (Punycode encoding = everyone.xn--q9jyb4c)
- 大坂.osaka (Punycode encoding = xn--uesx7b.osaka)
- みんな.大坂 (Punycode encoding = xn--q9jyb4c.xn--uesx7b)

Email Addresses using gTLDs and IDNs

Email addresses contain contains two parts: a local portion (the user name, to the left of the @ character) and a domain (to the right of the @ character). The domain portion can contain any TLD, including a gTLD. Both portions may be internationalized (an IDN).

Examples of Email Addresses using gTLDs and IDNs

- user@everyone.lawyer (uses new gTLD)
- user@everyone.みんな (uses international TLD)
- user@大坂.osaka (uses international 2nd level domain)
- 用戸@everyone.lawyer (uses international user name)

NOTE: An additional format, IDN Email Addresses, will be discussed below.

Universal Acceptance in Action

Defining the criteria

As mentioned above, Universal Acceptance is

> *The ability to Accept, Store, Process and Display all Top Level Domains equally and all IDNs, IRIs and email addresses equally*

These 4 criteria are described below.

- *Accept* – Applications and services may allow domain names and email addresses to be entered into user interfaces and/or received from other applications and services via APIs. Usually this process includes a validation to verify that the data has been presented in a valid format. Meeting this criterion depends on the application or service being aware of current valid formats, which include different lengths and character sets than was the case previously.

- *Store* – Applications and services may require long-term and/or transient storage of domain names and email addresses. Regardless of the lifetime of the data, it must be stored either in RFC-defined formats, or in proprietary formats which can be easily translated to and from RFC-defined formats. If a transformation must occur, it is a best practice for the transformation to occur as part of the storage system and not part of the acceptance system (i.e. storage in non-RFC-compatible formats should not be visible to the source of the data, whether that source is a human user of the UI or a binary source using an API.) Since multiple valid RFC-compatible formats may be involved in the formation of domain names and email addresses (e.g. ASCII, Punycode UTF-7 or Unicode UTF-8), transformation between valid formats may occur here
- *Process* – Processing means using domain names and email strings in a feature. There is no limit to then number of ways that domain names and email addresses could be processed (e.g. "identify all the people in New Zealand because they have a .nz name", "identify all the pharmacists because they have an @pharmacist email address".) Or Firewall's that might filter DNS requests that don't apply to their conventions.
- *Display* - When to display, when to do the transformation, whether to display based on the ability of the device or its settings. (If my phone only has ASCII and Chinese character sets enabled, what does it do when it wants to display Arabic?

Commented [MSv1]: This is actually a best practice, not a requirement, IMO. Does it belong here or elsewhere in the developer sections?

User Scenarios

The examples and definitions above may give the impression that Universal Acceptance is only about computer systems and online services. It's actually about real people using those systems and services.

Here are some examples of activities which may require Universal Acceptance.

- *Registering a gTLD* – A user wants to use a browser to view the website of a retailer. The retailer previously had a .com domain, and has encountered spoofing in the past (people who register a similar .com name and hope to fool users); now the company has acquired a gTLD for their brand. Not only does the gTLD reinforce their brand identity, they own the gTLD; no one else can use it and users can be confident that they are accessing the correct site.
- *Accessing a gTLD* – A user accesses a web site which has a gTLD. Even though the gTLD is new, any browser the user wishes to use will display the web address correctly and access the site correctly.
- *Accessing an IDN* - A user types an address or clicks a link pointing to a web site which has an IDN. Even if the IDN uses characters different than the language settings on the user's computer, any browser the user wishes to use will display the web address correctly and access the site correctly.
- *Using an international email address* – A user has acquired multiple email addresses and associates them as aliases to a common email inbox. Even though some of the email addresses are EAI, the user can send and receive email with them regardless who they communicate with or what email service they use.

Technical Requirements for Internet Internationalization

High level requirements

An application or service which supports universal acceptance:

- Supports all domain names regardless of length or character set <define script> <what about deliberate exception policy>

- Allows entry of international characters (i.e. all Unicode codepoints) into all UI inputs
- Can correctly render all code points in Unicode strings
- Can correctly render right-to-left strings such as those in Arabic and Hebrew
- Can communicate data between applications and services in formats which support Unicode and are convertible to/from UTF-8
- Offers public APIs which support Unicode & UTF-8
- Offers private APIs which support Unicode & UTF-8 (these private APIs apply only to inter-service calls by the same vendor)
- Stores user data in formats which support Unicode and is convertible to/from UTF-8 (such conversions would be visible only to the product/service owner)
- Supports all domain name strings in the authoritative ICANN TLD list and the community-served Public Suffix List regardless of length or character set
- Can send email to recipients regardless of domain or character set
- Can receive email from senders regardless of domain or character set
- Supports accounts which are associated with both an ASCII and Unicode email address aliases

<NOTE: The requirements and scenarios should probably contain some cross-references to the RFCs. Not to overwhelm the reader, just to add some context.>

Developer Considerations

Developing and Maintaining Universal Acceptance SW

Since many existing software systems contain hardcoded assumptions about domains and email addresses, code changes may be required. Developers should ensure that their programming environment offers libraries for the following features.

- Consuming an authoritative online resource to determine if addresses are valid
- Converting Unicode to/from ASCII (punycode)
- Converting UTF-7 to UTF-8
- Comparing strings in multiple formats
- Determining which script comprises a string
- Determining if a string contains a mix of scripts
- Recognizing homographs (characters which appear identical when rendered but which are distinct codepoints)

IDN Email

Because IDNs can be Punycode encoded, some existing software allows the IDN portion of an email address to be represented in ASCII or Unicode.

For example, in this method these addresses are equivalent:

- `user@everyone.みんな`
- `user@everyone.xn--q9jyb4c`

EAI is defined as using Unicode only; Punycode is not allowed. Nevertheless developers have sometimes adapted email software and services to handle IDN email addresses rather than make a full conversion to Unicode. Universal Acceptance software and services should be able to handle IDN email and convert it successfully to Unicode EAI format.

Best Practices

- Use authoritative resources to validate domain names. Do not make heuristic assumptions, such as “all TLDs are 2, 3, 4, or 6 characters in length.”
- Use Unicode-compliant APIs.
- Recognize that mixed-script is going to become more common. Don’t assume that mixed-script strings are intended for malicious purposes, such as phishing, and if your user interface calls such strings to the user’s attention, be sure that it does so in a way which is not prejudicial to user of non-Latin scripts.
- Don’t generate IDN email addresses, but be able to handle them if presented by someone else’s software.
- Complex scripts
- “Joiners” - RFC 5894, 4.3. Linguistic Expectations: Ligatures, Digraphs, and Alternat
- Case sensitivity
- I’m not sure what to say about these

Examples of Non-Universal Acceptance

asdasdsa

Glossary and other resources

Cross-link to RFCs

- RFC 3492 (Punycode)
- RFC 5890-94 (IDN)
- RFC 6530-33 (EAI)
- ISO 10646 (Unicode)
- GB18030 (China)

Online resources

- Unicode<>Punycode conversion libraries
- Windows APIs
- Sharepoint APIs
- Public Suffix List
- ICANN Authoritative TLD list
- Android & iOS APIs

Work to do

One of the other issues around UA readiness is TLD validation.

Can you include a chapter on that?

Also, It would be good to cover the four criteria for UA Readiness:

Accept - This includes validation

Store - This includes the ability to store the information - whether in Punycode UTF-7 or Unicode UTF-8. And if storing in 7 bit ASCII, when does the transformation take place.

Process - This should cover processing that might make use of a domain name. Finding all the people in New Zealand because they have a .nz name, for example. Or all the Pharmacists. Or Firewall's that might filter DNS requests that don't apply to their conventions.

Display - When to display, when to do the transformation, whether to display based on the ability of the device or its settings. (If my phone only has ASCII and Chinese character sets enabled, what does it do when it wants to display Arabic?

The issues of Arabic and Hebrew also should be covered.