Universal Acceptance Quick Guide

ACCEPT

VALIDATE

STORE

PROCESS

DISPLAY

Software and online services support Universal Acceptance when they offer the capabilities listed above for all domains and email names.

What Does "Universal Acceptance" Mean?

Some software does not recognize or correctly process all domain names and all email addresses. Domain names can include strings in the top-level position that are longer than the older familiar ones, and domain names and email addresses can now use characters drawn from a much larger Unicode-based repertoire than traditional ASCII[1]. **Universal Acceptance** (UA) is the state in which all valid domain names and email addresses are **accepted**, **validated**, **stored**, **processed**, and **displayed** correctly and consistently.

The Universal Acceptance Steering Group (UASG) is a community-led initiative working on creating awareness and identifying and resolving problems associated with the universal acceptance of all domain names and email addresses. Its goal is to help ensure a consistent and positive experience for Internet users globally. It is supported by ICANN (the Internet Corporation for Assigned Names and Numbers) and has participants from more than 200 organisations around the world, including Afilias, Apple, CNNIC, GoDaddy, Google, Microsoft, and Verisign. For more information on the UASG and recent developments, visit: **www.uasg.tech**.

This Quick Guide describes the UASG Recommendations for achieving Universal Acceptance in the five contexts—Accept, Validate, Store, Process, and Display—in which systems encounter domain names and email addresses. It is intended for executives and managers responsible for Information Technology or Software Product Engineering activities. It presents the UASG Recommendations at a high level without some of the detail that would be important to a software architect or engineer. For those details consult UASG 007, "Introduction to Universal Acceptance."

[1]ASCII is the character encoding historically used on the Internet, defined in the Internet standard RFC 20 (https://tools.ietf.org/html/rfc20). Unicode is defined by the Unicode Consortium (http://unicode.org).

ACCEPT

Accept is the process by which a domain name or an email address is received from a user interface, file, or API (application program interface) to be used by a software application or online service.

UASG Recommendations

- Input fields should be large enough to accept any valid input. Depending on how it is encoded, a domain name can require as many as 670 bytes. An email address can have a local part (the part before the @-sign) of up to 64 bytes in addition to a domain name, for a total length of up to 735 bytes.
- Applications and services should accept UTF-8[2] encoded domain names and email addresses, and should recognize that the number of bytes occupied by the UTF-8 encoding may be greater than the number of displayed characters.
- An IDN can be entered and displayed either in its original script or in an ASCII version designed for backward compatibility; for example 测试 and xn--0zwm56d. The Unicode encoding of the original script is called a U-label[3]; the equivalent ASCII-compatible encoding is called an A-label. Every A-label corresponds to one U-label and vice-versa. Software should accept both A-labels and U-labels, but convert A-labels to U-labels for display and for any processing that does not require A-labels.
- In almost all cases an entered domain name or email address should be converted into Unicode Normalization Form C (NFC)[4] before further processing. Because NFC is not perfectly lossless, in rare circumstances it may be necessary to defer normalization until further processing has established the specific context(s) in which it should be applied.

[2]UTF-8 encodes each Unicode code point as a sequence of one to four bytes. It is defined in RFC 3629.

[3]Conversion between U-labels and A-labels is accomplished by the "Punycode" algorithm defined in RFC 3492 and RFC 5891.

[4]See Unicode Standard Annex #15, "Unicode Normalization Forms" (https://www.unicode.org/reports/tr15/tr15-47.html).

VALIDATE

Validate is the process to check an email address or domain name for correct syntax and, when appropriate, that a name that is expected to exist in the DNS actually does. Validation techniques may need updating to work with modern domain names and email addresses.

UASG Recommendations

- Input should be validated in a manner appropriate for its intended use. All domain names should be validated against the Internationalized Domain Names in Applications standard, currently IDNA2008[5]. This ensures that the name is syntactically valid.
- If an input string is expected to be an existing entry in the DNS, validate it with a DNS lookup.
- If an input string is expected to be a valid domain name that might not (yet) be in the DNS, it may still be possible to validate part of it. For example, the top-level domain (TLD) name can be checked against the authoritative list of valid TLD names maintained by the Internet Assigned Numbers Authority (IANA)[6].
- To validate an email address, validate the domain part as described above. Because the local part of an email address is defined only by the mail system that receives mail, it is generally not possible to validate it. Asking the user to enter the email address twice may detect typing errors.
- In most cases, all of the components of a domain name or email address (except the TLD name if it is not an IDN) should be in a single script (e.g., Arabic or Han) or closely related scripts (e.g., Japanese Kanji, Katakana, Hiragana, and Romaji). Use Unicode Technical Standard #39, "Unicode Security Mechanisms (https://www.unicode.org/reports/tr39/#Restriction_Level_Detection), to check that the scripts in a Unicode sequence follow good practice.

[5]See IETF RFCs 5890, 5891, 5892, 5893, and 5894 for the definition of IDNA2008.

[6]See "List of Top-Level Domains" (https://www.icann.org/resources/pages/tlds-2012-02-25-en).

STORE

Store refers to the temporary or long-term storage of domain names and email addresses, which should be stored in well-defined formats regardless of the expected duration of the storage.

UASG Recommendations

- In almost all cases domain names and email addresses should be normalized according to Unicode Normalization Form C (NFC) before storing. Because NFC is not perfectly lossless, in rare circumstances it may be necessary to defer normalization until further processing has established the specific context(s) in which it should be applied.

- In most applications, domain names and email addresses should be stored in files and databases encoded as UTF-8, the most common and best-supported Unicode encoding. In some cases, where software has to interoperate with legacy databases, it may be easier to use the same encoding as the database.

- Within application code, the most appropriate representation of Unicode depends on the programming environment. Many common programming languages including the python and perl scripting languages have built-in support for Unicode and automatic conversion to or from UTF-8 on input and output.

- Applications should choose a consistent internal representation—either U-labels or A-labels—for IDNs. Because every U-label can be transformed into a unique A-label and vice-versa, either form is acceptable.

PROCESS

Processing occurs whenever an email address or domain name is used by an application or service to perform an activity (e.g., searching or sorting a list) or changed into an alternative format (e.g., from a legacy encoding into UTF-8). Additional validation may occur during processing.

UASG Recommendations

- As Unicode evolves, upgrade software when practical to use the most recent version of the standard and any available graphics and fonts. Consider that user devices, software libraries, and web standards may not support the most recent version, and therefore may display newly allocated characters incorrectly, as a generic box ( ), or not at all.
- When APIs that support UTF-8 input or output are available, use them rather than APIs that do not. Use standard well-debugged libraries, such as the GNU libidn2 (https://www.gnu.org/software/libidn/#libidn2), to process and validate IDNs; do not "roll your own."
- Scripts that are written right-to-left require special considerations when they are used in domain names and email addresses. Some of those considerations are addressed in IDNA[7] (for domain names) and an Annex to the Unicode Standard[8] (for email addresses).
- When creating registries or other data structures that include script or language information, allow for as many as possible, ideally all that the Unicode Standard supports[9]. Be aware that some languages can be written using different scripts, and that some scripts can be used to write many different languages.

[7]See RFC 5893, "Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)" (https://tools.ietf.org/html/rfc5893).

[8]See UAX#9, "Unicode Bidirectional Algorithm" (http://unicode.org/reports/tr9).

[9]See the Unicode "Supported Scripts" (http://unicode.org/standard/supported.html).

DISPLAY

Display occurs whenever an email address or domain name is rendered visually by a user interface. Displaying domain names and email addresses is usually straightforward when the scripts used and any required rendering mechanisms are supported in the underlying operating system and strings are stored in an encoding defined by the Unicode Standard. Application-specific transformations may be required otherwise.

UASG Recommendations

- Consider that although modern software and devices can display nearly all Unicode code points, older systems may have limited support, and require that applications manage some of their old fonts. Also, when Unicode adds new code points, devices and applications will not display them until their font libraries have been updated.
- Display IDNs in their native character form unless there is a specific requirement to display them as A-labels.
- Domain names and email addresses can be displayed in left-to-right (LTR) text, as in English or Russian, or right-to-left (RTL) text, as in Arabic or Hebrew. Because Unicode assigns directionality attributes to individual code points—not to code point sequences—some mixed LTR and RTL ("bidirectional") text makes sense to users, and some does not. Use the Unicode restriction levels criteria[10] to flag potentially misleading strings.
- Internet users read and speak many different languages. In some cases it may be necessary to design applications separately for different languages or language groups.

[10]See Unicode Technical Standard #39, "Unicode Security Mechanisms" (https://www.unicode.org/reports/tr39/#Restriction_Level_Detection), for its Moderately Restrictive and Highly Restrictive restriction levels to check that the scripts in a Unicode sequence follow good practice.

Become Universal Acceptance-Ready

Source Code Reviews & Unit Testing

Software and systems that have been developed or upgraded to support Universal Acceptance should be reviewed and tested to ensure correctness and to find and fix bugs. As part of the UASG awareness efforts, the group is reaching out to application developers and online service providers to encourage them to perform universal acceptance source code reviews and testing and share a list of criteria which can be used to develop standardized test cases.

Testing

UASG is also developing a list of web sites, applications, email addresses, and domain names usable for testing. In some cases, tests can be automated and run without manual intervention. A real-world example is the recent gTLD investigation performed by APNIC Labs on behalf of ICANN: https://tinyurl.com/new-gtld-ua. UASG is investigating methods of automated testing for universal acceptance and will share findings as they are available.

Further Reading

These documents have further information on Universal Acceptance, Unicode, and Internationalized Domain Names.

- UASG 007, "Introduction to Universal Acceptance" (https://uasg.tech/documents).
- RFC 5894, "Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale" (https://www.rfc-editor.org/info/rfc5894).
- "International typography on the Web," a graphical summary showing problems and issues handling different languages on the Web (https://w3c.github.io/typography/gap-analysis/language-matrix.html).

A note on terminology

One of the difficulties with achieving Universal Acceptance is that many terms and concepts that are familiar to people accustomed to simple scripts with a small number of distinct "alphabetic" characters, such as the Latin script, can lead to a great deal of confusion when applied to writing systems that utilize different principles. Incorporating a wide variety of writing systems into the field of internationalized domain names (IDNs) has required the invention of new terms and the use of familiar terms (such as "character") in special and very specific ways. This Quick Guide tries to avoid such terms or to define them when they are used, but an examination of other materials, including some of the documents referenced here, is likely to require a deeper understanding of the terminology.