Universal Acceptance (UA) Roadmap for Domain Name Registry and Registrar Systems

11 January 2023

Table of Contents

1 Executive Summary

Universal Acceptance (UA) is the ability for applications and infrastructure to properly support internationalized domain names (IDN), internationalized email addresses (EAI) and top-level domain (TLD) names of various lengths and types. Software applications have to be built or updated to support those new capabilities. These include applications that are used in the registration of domain names, such as domain name registry and registrar software systems. This document explores a methodology to address the UA-readiness of those systems for domain name registry and registrar operators, registry backend providers, developers, and technical managers.

The work defines gates within these applications to apply UA related changes and to test UA-readiness. This work applies to generic top-level domain registry and registrar systems, and may be applicable to certain country code top-level domains (ccTLDs).

Protocols and interfaces to the protocols involved in the registration ecosystem, such as Extensible Provisioning Protocol (EPP), WHOIS and Registration Data Access Protocol (RDAP) are also addressed. Finally, considerations for properly testing such systems are detailed, including appendices which share testing results of actual registry and registrar systems. As this document does not specifically address IDN variant labels, which have a major impact on these systems, some considerations are discussed to prepare these systems to support IDN variant labels. Specifics also include the requirements for the ICANN gTLD environment, such as mandatory reports and data escrow.

2 Introduction and Scope

Universal Acceptance (UA) is the ability for applications and infrastructure to properly support internationalized domain names (IDN), internationalized email addresses (EAI) and top-level domain (TLD) names of various lengths and types [UASG026]. In this document, UA objects are used to encompass all of these internationalized identifiers. This document is of technical nature and targeted to registry and registrar operators, registry backend providers, developers and technical managers.

The document suggests a roadmap for domain registries and registrars to follow in order to fully support UA in their systems, applications, and infrastructure. This is a follow-up work on *The Role of Country Code Top-Level Domains (ccTLDs) in Achieving Universal Acceptance* [APTLDREPORT] published by Asia Pacific TLD Association (APTLD). The UA roadmap is a technical study serving the purposes of raising awareness of UA and to assist relevant parties with becoming UA-ready.

To achieve this goal, two registry systems, Google's open-source Nomulus [NOMULUS] and Knipp's TANGO [TANGO], and COREhub's registrar system [GATEWAYNG] were reviewed. This study also reviewed protocols involved in the domain registration systems.

This study focuses on systems used by ICANN-accredited registries and registrars (contracted parties) to manage domains under gTLDs, which includes various services and reporting specific to ICANN contracts. However, this study may generally apply to ccTLD registry and registrar environments as well.

This study only focuses on Universal Acceptance, i.e. IDN, EAI and TLDs of various types. Other internationalized data such as a registrant internationalized postal address is out of scope. Internal identifiers which are typically ASCII such as Repository Object Identifiers (ROIDs) are also out of scope.

3 High-Level Domain Name Registry and Registrar System Architectures

This section provides a high-level view of the domain name registry and registrar system architectures. The objective is to identify key features and gating locations for verifying UA-readiness. Implementations can be reasonably expected to vary from this architecture, as also discussed in more details below. Therefore, the implementer may need to adapt the roadmap. UA-readiness gates identified as G*x* are discussed later.

Figure 1 illustrates an example of a domain name registry high-level system architecture.

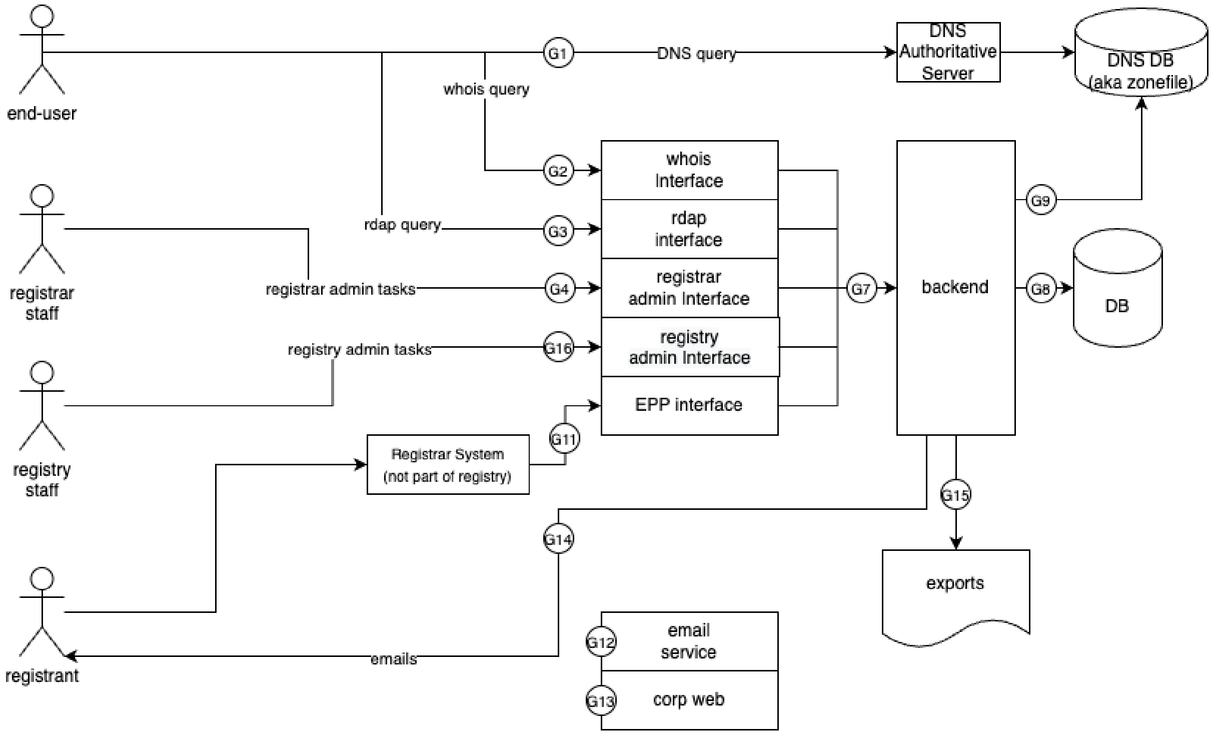

Figure 1: Domain Name Registry High-Level System Architecture

For a domain name registry, there are primarily four different personas: the Internet end-user, the registrant, the registrar staff, and the registry staff. This following simple description of interactions should not be new for those involved in the industry. The purpose of listing these interactions is to identify gating points to verify universal acceptance-readiness.

The Internet end-user interacts with the registry system through Domain Name System (DNS) queries to its DNS resolver, querying for DNS records of the TLD managed by the registry. The end-user also queries registration data of domain names using either WHOIS or RDAP on the registry-related interfaces. End-user should be interpreted to encompass programs, bots, search engines, etc.

The registrant interacts with the registrar interface to do various actions related to the provisioning of a domain name under a TLD managed by the registry. The registrant may receive emails from the registry for some administrative tasks such as confirmation of a registrar change, required sign-up, registrant eligibility assessment, purchase etc. A potential registrant may sign-up for this purpose. However, this is not the case with all registries and varies between operating models.

The registrar and registry staff do various administrative tasks, such as user management, Internet Protocol (IP) whitelisting and sometimes domain management. In some cases, systems supporting these functions may also need to support UA.

The domain name registry has various corporate services such as a website and email service. The registry must also export various data to various entities such as ICANN to meet with the contractual requirements.

Figure 2 illustrates an example of a domain registrar high-level system architecture.

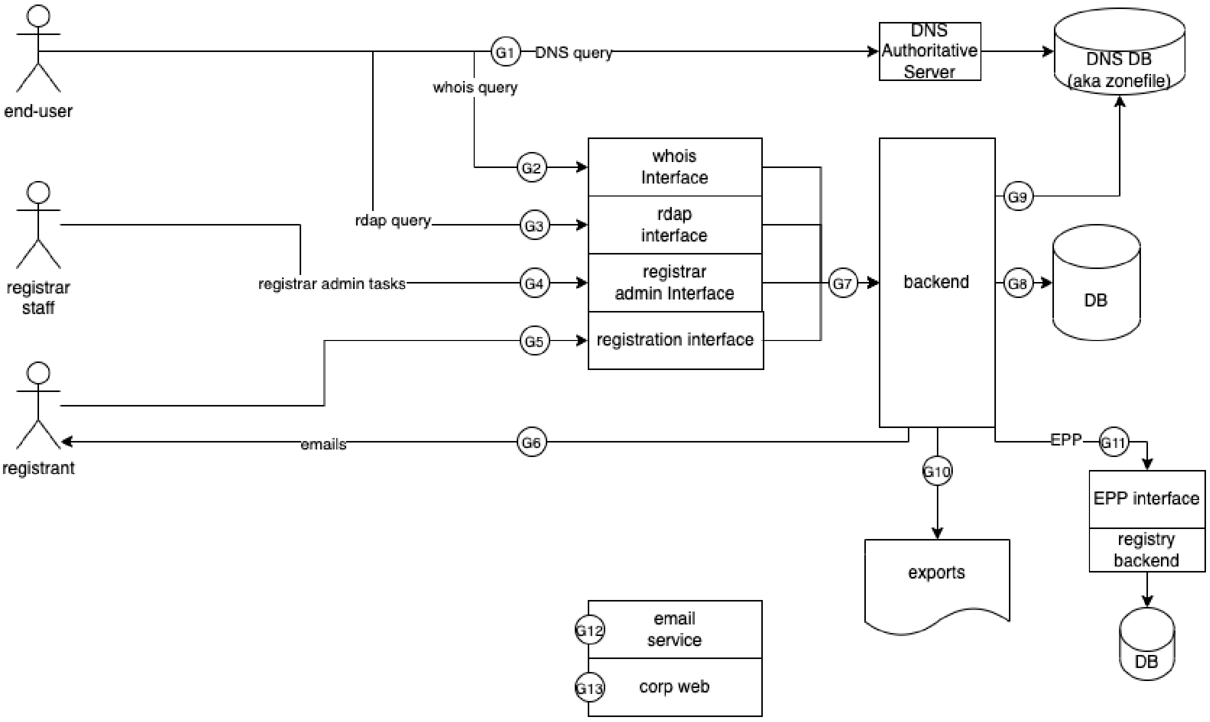

Figure 2: Domain Registrar High-Level System Architecture

For a domain registrar, there are primarily three different personas: the Internet end-user, the registrant, and the registrar staff.

The Internet end-user interacts with the registrar system through DNS queries to its DNS resolver, querying DNS records of a domain managed by the registrar DNS systems (typically the second-level domain). The end-user also queries registration data of domain names using either WHOIS or RDAP on the registrar related interfaces.

The registrant interacts with the registrar interface to perform various actions related to the provisioning of a domain name for TLDs managed by the registrar. The registrant may receive emails from the registrar for some administrative tasks such as the renewal of its domains. A potential registrant may sign-up for this purpose.

The registrar staff performs various administrative tasks. The interface may be a web interface, as an example.

The domain registrar has various corporate services such as a website and email service. It also has to export data and reports to external parties such as ICANN, the Trademark Clearinghouse, or an escrow agent in order to comply with the contractual requirements.

The domain registrar backend interacts with the registry backend through the EPP for all actions related to the provisioning of a domain and their related objects like hosts and contacts.

It should be noted that additional protocols and interfaces may be provided by the registrar or the registry. For example, other domain provisioning protocols than EPP are also used by some registries and registrars. Other interfaces such as Representational state transfer (REST) application programming interface (API) may also be used by some registry, registrars, resellers, and backend providers for various applications. Those additional protocols are not required by ICANN contracted parties. This study did not investigate those other protocols or interfaces for UA-readiness, so that work should be done by the relevant parties. However, the framework proposed in this document should be easily adaptable to those specific protocols and interfaces.

3.1 Exports and Reports

ICANN contracted parties (registries and registrars) typically have requirements under their ICANN contracts to provide data and reports to external entities. The data and reports may contain objects that may have UA-readiness considerations. For example, at the time of publication, certain ICANN contracts include requirements for the following exports/reports:

- Per-Registrar Transaction Reports
- Registry Functions Activity Reports
- Bulk Registration Data
- Zone Files
- Data Escrow Report
- Data Escrow (RFC8909 format)
- IDN Tables

- List of Registered Domain Names (LORDN) (draft-ietf-regext-tmch-func-spec format) (Trademark Clearing House, TMCH)
- Qualified Launch Program (QLP) Names List (TMCH)
- Additional Data Required by ICANN from Contracted Parties

4 Generic Application Architecture for UA-Readiness Verification

Universal Acceptance Readiness Framework [UASG026] proposes a framework for UA-readiness verification of applications, as illustrated in Figure 3. This section is a short summary of the framework.

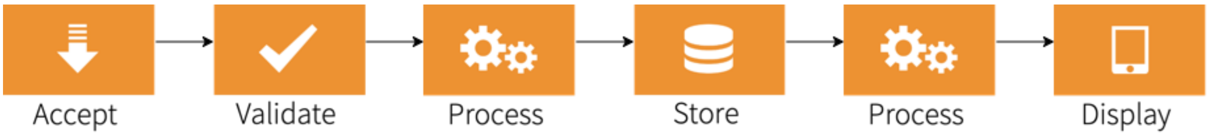

Figure 3: UA-Readiness Gates

In a typical application, a user enters a string (in the context of this study, an email address, a domain name, a nameserver host name, or perhaps another identifier).

1. The application must first “accept” the string and perform some superficial error checking. The application needs to accommodate the maximum length of the identifier and needs to reject unacceptable characters (e.g., ';'). The application should not limit the length of a TLD string to less than the maximum DNS label size (63 octets) in the DNS specification [RFC 1034]. For email addresses, it could verify the presence of the “@“ character and a minimum of one ‘.’ in the domain part.
2. Next step is to “validate” a label, e.g., for an IDN label it means verifying that it is valid as per the requirements of the IDNA2008 protocol, e.g., see [RFC5890].
3. The first “process” step processes the input to the application, where it should not, for example, convert a string to a type that does not accommodate internationalization, such as plain ASCII string types.
4. The “store” step typically involves a database where for example the field type should not lose any internationalization information of the strings.
5. Similarly, a system may require follow-up process in the application which would read the database and do some processing for output that should not lose any internationalization information.
6. Finally, the strings should be properly displayed on the user interface, which means properly converting the strings and also labeling them correctly.

A set of test cases is in the *Readiness Framework* report [UASG026]. This framework together with the generic software architecture of registrar and registry systems shown in the previous section are used in this report to identify UA-readiness gates and to suggest appropriate test strategies and test cases.

5 UA-Readiness Gates

Figure 1 and 2 display gates as G*x*, where *x* is a sequential integer. At these gates, proper testing of UA should be performed, depending on the functionality. Table 1 shows the spread of testing for each gate.

Table 1: UA-Readiness Gates

Gate #	Short Description	Accept	Validate	Process	Store	Process	Display
G1	DNS query			X			
G2	WHOIS query		X	X			
G3	RDAP query		X	X			
G4	registrar admin interface	X	X	X		X	X
G5	registrant interface	X	X	X		X	X
G6	emails sent by registrar to registrant			X		X	X
G7	backend		X	X			
G8	database				X		
G9	DNS zone file generation			X	X		
G10	exports and reports for registrar			X	X		
G11	EPP		X	X			
G12	corporate email			X		X	X
G13	corporate web	X	X	X		X	X
G14	emails sent by registry to registrant			X	X	X	X
G15	exports and reports for registry			X	X		
G16	registry admin interface	X	X	X		X	X

UASG026 report [UASG026] on page 9-10 lists the various tests for each of the functions: accept, validate, process, store, display. Any testing plan should be based on those tests.

The remainder of this section is about specific cases not necessarily covered by the UASG test plan [UASG026] or require additional considerations.

G1 is a DNS query by an Internet end-user. The input of the end-user may be an IDN in the form of a U-labels (i.e., using UTF-8) but the DNS query software may not correctly handle Internationalized Domain Names in Applications (IDNA)[RFC5890], such as the proper conversion to (or from) A-labels (in a response). But there is no action on the registry or registrar software.

G2 is a WHOIS query by an Internet end-user. The WHOIS [RFC3912] protocol may accept UTF-8 but there is no signaling within the protocol to identify the character set being used, therefore the registry and registrar WHOIS interfaces may receive queries for domains encoded with U-labels. The response may include both IDN and EAI, such as the email address of the registrant. Validation and processing of these domains should follow [UASG026] test plan.

G3 is a RDAP query by an Internet end-user. RDAP is designed to support internationalization. RDAP receivers should expect IDN in any form for a query and should be able to send EAI and IDN in their response. More details on RDAP fields appear in the following sections:

G4, G5, and G16 are typically web applications, therefore care should be taken for properly accepting, validating, processing, and displaying UA objects (EAI and IDN).

G6 and G14 are emails sent by backend system to registrants for various services. If that is the case, then it requires the backend system to properly format the content of the emails if the content contains UA objects, and if the destination email address is an EAI, to properly send the email through an EAI compliant Simple Mail Transfer Protocol (SMTP) proxy service and to properly catch any rejection of EAI transport through the SMTP path. For an EAI email to properly arrive at its destination, each mail server in the path must support EAI, which means for example, advertising the SMTPUTF8 option in the initial handshake.

G7 is the backend that does most of the processing, therefore deeply involved in processing IDN and EAI. It should be noted that programming languages and libraries have various levels of support for those UA objects, from zero support to full support. A comprehensive study and test of most used programming languages and frameworks is available in a detailed report [UASG037]. UA-ready code samples for various languages and frameworks are also available from the UASG site [UASG].

G8 is the database. The database schema should support UTF-8 for UA objects. Normalization should be done for any UA object before storing and search tokens should be pre-normalized before hitting the database search engine. Storing IDNs in UTF-8 is not necessary except if search and sorting is used. When search and sorting is used, the UTF-8 version of an UA object should be the key of the indexes.

G9 is the DNS zone file generation, which involves converting all U-labels into A-labels before storing the zone file.

G10 and G15 are the exports and reports provided to external entities such as ICANN. Table 2 lists the important fields for UA-readiness.

G11 is the EPP protocol. EPP uses Extensible Markup Language (XML) which theoretically means it supports UTF-8. However, EPP was defined before Internationalizing Domain Names in Applications (IDNA), so these interfaces are not expecting domain names or email addresses to be non-ASCII. Therefore, all domain names in EPP should be encoded as A-labels. If EAI email addresses are to be used, the new EPP extension for EAI [EPPEAI] should be used by the sender and the receiver.

G12 and G13 are generic and include a variety of applications and systems, such as customer relationship management, communications, ticketing. If applicable to a registry system, these should be investigated the same way as proposed in this report, such as identifying gates and testing UA-readiness.

G12, G13, G14 and G16 may not be used by or applicable for some entities.

5.1 Exports and Reports

Table 2 lists the export and report types with the relevant UA field(s).

Table 2: UA-Readiness of Exports

Export Type	Export Name	IDNA	EAI
Monthly reports	Transaction	TLD in each row	
	Activity	TLD in each row	
	Spec 11	fullyQualifiedDomainName in Threat json object	registrarEmailAddress in threatMatches json object
	LORDN (TMCH)	domain column	
BRDA		tld, domain, name/unicode_name (NNDN)	
Zone file access		DNS records	SOA record

Registry deposits	Escrow	rdeDom:name, rdeHost:name, rdeRegistrar:url, rdeRegistrar:WHOISInfo, rdeNNDN:aName, rdeNNDN:originalName rdeNNDN:uName, rdeDomain:uName, rdeDomain:name, domain:hostObj, rdeDomain:originalName, rdeHost:name, rdeHeader:rcdn	rdeContact:email, rdeRegistrar:email
	Report	rdeHeader:tld	

UA issues arising in exports and reports relate mainly to the processing and storage of domain names and email addresses. A standard way of doing exports is to extract the information from the database (by SQL database queries in the case of Nomulus for instance) and process the export format. UA elements that should receive special attention are listed in the table above.

Of course, the provider database must support the storage of EAI and IDN as described earlier.

Every domain name (TLD, fullyQualifiedDomainName, rdeNNDN:aName, etc.) involved during an export should be exported as an A-label. If the database string is the U-label form, it must then be converted to an A-label beforehand.

There are only 2 exceptions to this rule where the U-label form is expected and must be preserved:

- BRDA: unicode_name
- RDE: uName

6 Protocols

For protocols such as EPP, RDAP, and WHOIS, the following table lists the elements related to UA objects. A test plan should contain test cases for each of these elements.

Table 3: Protocols UA Fields

Protocol	Schema Elements (xsd or json-schema)	IDN	EAI
EPP	eppcom:labelType	X	
	eppcom:minTokenType, contact:discloseType	X	X
RDAP	domainName (ldhName or unicodeName)	X	

	uri	X	
	email (jcard/vcard)	X	X
WHOIS	Registrar WHOIS Server	X	
	Registrar URL	X	
	Domain Name	X	
	Registrar Abuse Contact Email	X	X
	Registrant Email	X	X
	Admin Email	X	X
	Tech Email	X	X
	Email	X	X

Since the EPP protocol [STD69] was created before IDNA, domain names and hostnames labels must be in A-label formats to ensure full compatibility with any EPP parser. A new extension to the EPP protocol to support EAI [EAIEPP] is being developed by the Internet Engineering Task Force (IETF).

WHOIS is a binary protocol with no tagging of character set whatsoever. This can cause issues with the local part of an email address that would include non-ASCII characters. However, if the local part is ASCII but the domain is not, the domain could be converted to an A-label to partially support EAI. This mitigation should be used also for emails in EPP (eppcom:minTokenType schema elements).

The following table lists the functions using each of the schema elements in the protocols above.

Table 4: Schema elements of protocols

Schema Elements (xsd or json-schema)	Functions
eppcom:labelType	Check, Create, Delete, Info, Renew, Transfer, Update
eppcom:minTokenType,	Create, Info, Update
domainName (ldhName or unicodeName)	Domain, Nameserver, Entity (through port43 subschema)
uri	Nameserver, Domain, Entity, Help (all through link subschema)
email (jcard/vcard)	Entity (through vcardArray subschema)
Registrar WHOIS Server	Domain_lookup, Registrar_lookup, Nameserver_lookup

Registrar URL	Domain_lookup, Registrar_lookup, Nameserver_lookup
Domain Name	Domain_lookup
Registrar Abuse Contact Email	Domain_lookup
Registrant Email	Domain_lookup
Admin Email	Domain_lookup
Tech Email	Domain_lookup
Email	Registrar_lookup

Except for the RDAP unicodeName json element and the Port43 WHOIS fields “Internationalized Domain Name” and “Internationalized Server Name”, all other elements should use the A-label form of the domain.

7 Considerations

7.1 UA-Readiness Project Planning

Depending on the complexity of software systems a registry or registrar has, it may be appropriate to plan for UA-readiness in multiple phases. A typical phased approach would be first to review all systems and services for UA-readiness and identify the ones that need work. This would also include reviewing languages and libraries to verify their own UA-readiness, using UASG report [UASG037A]. Second, identify a list of systems to update and plan for pilots and testing. ICANN org implemented a phased approach to make its systems UA-ready [UASG013] where three stages were identified:

1. support for new short and long ASCII-based TLDs
2. support for IDNs
3. support for EAI

Further details are described in the report. Each organization will need to adapt this case to its own environment. For example, it may be more efficient to collapse some of these steps into a small set.

7.2 System Tests

As with any application, unit tests and system tests are complementary to each other and both should be exercised. For example, a round trip test case may involve entering a string in the registration interface (G5) and then verifying if it displays properly within that registration interface after being saved in the database, but also should be verified in the various other places in the system such as when sending emails (G6), when exporting reports (G10), when queried and answered in WHOIS (G2) and RDAP (G3) queries, etc. Similarly, all gates should be exercised in the system by providing various test inputs and verifying that the other gates are properly handling those strings.

7.3 Third-Party Library Usage

Modern software development uses a large set of open-source libraries. For example, an open-source registry system, developed in Java, uses the following libraries:

- com.google.guava;
- javax.mail;
- com.ibm.icu;
- java.net.IDN;
- com.google.http-client:google-http-client;
- org.apache.httpcomponents:httpclient;
- java.net.URL

Libraries dealing with domain names or email addresses may not be UA-ready, as shown in *Readiness of Some Programming Language Libraries and Frameworks* [UASG037]. As noted in that report, not all libraries listed above properly support UA. Therefore, code should be added, usually in front of the call to the library routine, to properly support UA, or use another (or updated) library which is UA-ready. The latter may be more difficult as the non-UA functionalities may be more useful for the developer. Adding wrapping code to support UA and how to properly use libraries is described in a report “UA-Ready Code Samples in Java, Python, and JavaScript” [UASG043] which also has code samples for most commonly used libraries.

7.4 Normalization

Some characters can be represented in multiple bytecode representations in Unicode. For example, “é” can be represented as a single codepoint in a composed form (“é”: U+00E9 or as two codepoints in a decomposed form (“e” +”́” : U+0065 + U+0301). Therefore, comparing, searching or sorting requires normalizing the strings encoded in Unicode in the decomposed form (NFD) or composed form (NFC) before doing the operation. Most Unicode-aware libraries typically normalize to compare. Moreover, IDNA [RFC5890] requires normalization (NFC). However, in some cases, depending on where the string comes from, it may not be normalized. Therefore, a good practice is to enforce normalization (NFC is suggested) on strings coming from outside before processing or storing it. The test plan should include specific test cases with unnormalized strings.

7.5 Different Script Test Cases

In a Unicode-based application, there should be no issue regarding the use of some languages or scripts, as Unicode encompasses all scripts and languages. However, there are a few considerations that sometimes create issues. The main consideration is related to display.

A system may support many languages and scripts, and different scripts handle codepoints differently especially regarding diacritics. Thus, it is appropriate to verify the proper display of various structurally different scripts, including alphabetical scripts, such as Cyrillic and Latin, abugida scripts, such as Ethiopic, Tamil or Thai, and right-to-left (RTL) scripts, such as Arabic or Hebrew. Moreover, it is important to also verify when scripts are mixed either within a label, when possible, or where each label of a domain name is from a structurally different script, such as one label in alphabetical script and the other in an RTL script.

7.6 Test Data

UA-readiness testing should consider using the list of compiled strings *Test Cases for UA Readiness Evaluation* [UASG004, UASG004A] done by the UASG. This list is comprehensive and provides the best coverage for various use cases. These test cases should be integrated in the continuous integration/continuous delivery pipeline. If the list is too comprehensive and manual testing is done, then choose a smaller set based on the criteria in the section above.

To fully test EAI conformance, it is necessary to set up an email server instance supporting EAI since the SMTP dialog is modified for EAI. Properly supporting EAI not only means supporting EAI address in fields and database, but also properly sending email using the SMTP extensions for EAI and properly recovering issues when sending or when the email cannot reach its destination.

7.7 IDN Handling

The IDNA protocol [RFC5890] defines whether a Unicode code point is valid or not for an IDN. Further restrictions are applied by additional policies.

For example, IDN top-level domain (TLD) labels must be compliant with the Root Zone Label Generation Rules [RZ-LGR] which further restricts the repertoire and rules for IDN TLD labels. For future gTLDs, this statement reflects the recommendation 25.2 of the Subsequent Procedures Final Report adopted by the Generic Name Supporting Organization (GNSO) Council [SubPro]. For existing TLDs, this statement is dependent on the outcomes of the GNSO IDN Expedited Policy Development Process (EPDP). For IDN ccTLDs, it would follow the outcome of country code Name Supporting Organization (ccNSO) policy development process (ccPDP4).

At the second level, IDN tables are defined for each TLD as published by IANA [IDNTABLES]. Therefore, further validation is necessary for registries' and registrars' software to verify if a domain name is valid by applying those additional rules. It should be noted that applying LGR [RFC7940] rules is complex. At this time of writing, only one library and tool written in Python is available that fully implements LGR processing [LGRCORE, LGRDJANGO]. An instance of that tool is available and running at [LGRTOOL]. Implementers should be advised that developing a new LGR processing library is a significant undertaking and using these tools can dramatically reduce the development time.

8 Test Plan

A test plan should be structured to accomplish the previously discussed considerations, by having test cases:

- using IDN and EAI test data
- in various scripts, e.g., those supported by [RZ-LGR].
- with normalized and unnormalized strings
- applied to all UA-readiness gates
- on specific fields for protocols
- using proper libraries

An example of such a test plan executed on an actual registry and registrar systems is available as separate appendix documents (see links in the appendices below).

9 IDN Variant Labels

IDN variant labels are different labels that are grouped together because they are deemed equivalent under relevant label generation rules. Each label could either be allocatable for registration or blocked from registration.

The IDN repertoire and variant code points for the DNS root zone are defined by the Root Zone Label Generation Rules published by ICANN [RZLGR). IDN repertoires and variant code points at the second level are defined and submitted by the registries to IANA and are located at IANA IDN tables [IANAIDNTABLES]. IDN variants for each language and script supported by registries are defined using either one of the two historical formats [RFC 3743, RFC4290] or by its Label Generation Rules [RFC7940] format, the now recommended format.

Variant code points are defined for single characters or sequences of characters, although the latter is rare. There may be multiple variant code points for a single character, in which case it is a variant set. Therefore, a single label may have multiple characters that have variant sets, defining a large set of variant labels. Since a typical domain name has two labels, the TLD and the second level, and each label may have multiple variants, the expanded set of domain variants can be significant. Some scripts may have so many variant code points for a codepoint that calculating all the variant labels of a label may be computationally very expensive.

In various phases of domain registration, the impact of variant labels varies. For example, a sunrise period (i.e., the phase the registry starts or introduces some new registrations) should be cognizant of variant labels. IDN variants at the DNS root zone level and at second level should belong to the same registrant as discussed in the IDN Variant Implementation Guide [IDNVARIANTIMPLGUIDE].

Investigation of the impacts of variant labels has not been done in this study due to their dependency on the policy development processes currently underway by GNSO and ccNSO. However, the readers should be aware that IDN variant labels do have very significant impacts on domain registration applications, in all parts of the application. Essentially, all gates defined in Figures 1 and 2 are impacted by the use of variant labels.

10 Conclusion

Universal Acceptance is essential for the continued expansion of the Internet and provides a gateway to the next billion Internet users. It ensures that all domain names and email addresses can be used by all Internet-enabled applications, devices, and systems. It enables the private sector, government, and societies to better serve their communities through the use of an increasing number of new domains, including non-Latin based, language-specific IDNs. This will enable millions of users to navigate the Internet via new gTLDs in their native scripts and languages. UA is particularly relevant and beneficial for registries and registrars supporting domain names regardless of length, language, or script.

Supporting Universal Acceptance in registry and registrar systems requires identifying the gates where UA specific tests are applied and to define and use the right set of test cases. This document provides the relevant information that is needed for registries and registrars to become UA-ready and leverage the opportunities that all new gTLDs provide.

11 Acknowledgements

(in alphabetical order)

- Consultants:
  - Michael Bauland
  - Guillaume Blanchet
  - Marc Blanchet

- ICANN Staff:
  - Seda Akbulut
  - Kyle Dunst
  - Sarmad Hussain
  - Edward Lewis
  - Mert Saka
  - Champika Wijayatunga

Appendix A - Registry Testing

This appendix covers examples of testing of the gates identified earlier in this document for the registry system. It serves as an approach to test the registry's main features that are easily accessible without internal knowledge. As such we are providing sample test cases for the UA-readiness gates G1, G2, G3, G4, G11, and G15, thereby implicitly also testing the internal gates G7, G8, and G9.

Appendix A is available at this link.

Appendix B - Registrar Testing

This appendix covers examples of testing of the gates identified earlier in the main document for the registrar system. It serves as an approach to test the registrar's main features that are easily accessible without internal knowledge. As such we are providing sample test cases for the UA-readiness gates G1, G2, G3, G4, G5, and G6 thereby implicitly also testing the internal gates G7, G8, G9, and G11.

Appendix B is available at this link.

References

[APTLDREPORT] The Role of Country Code Top-Level Domains (ccTLDs) in Achieving Universal Acceptance, APTLD, July 2021, https://aptld.org/documents/working-papers/17880/
[EAIEPP] Use of Internationalized Email Addresses in the Extensible Provisioning Protocol (EPP), D. Belyavskiy and J. Gould, May 2022, https://datatracker.ietf.org/doc/html/draft-ietf-regext-epp-eai
[IDNVARIANTIMPLGUIDE] https://www.icann.org/resources/pages/idn-variant-tld-implementation-2018-07-26-en
[IANAIDNTABLES] https://www.iana.org/domains/idn-tables
[GATEWAYNG] https://corehub.net/services/registrar-platform/
[LGRCORE] https://github.com/icann/lgr-core
[LGRDJANGO] https://github.com/icann/lgr-django
[LGRTOOL] https://lgrtool.icann.org
[NOMULUS] https://google.github.io/nomulus/
[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, https://www.rfc-editor.org/info/rfc1034.
[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, https://www.rfc-editor.org/info/rfc1123.
[RFC3743] Konishi, K., Huang, K., Qian, H., and Y. Ko, "Joint Engineering Team (JET) Guidelines for Internationalized Domain Names (IDN) Registration and Administration for Chinese, Japanese, and Korean", RFC 3743, DOI 10.17487/RFC3743, April 2004, https://www.rfc-editor.org/info/rfc3743.
[RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, DOI 10.17487/RFC3912, September 2004, https://www.rfc-editor.org/info/rfc3912.
[RFC4290] Klensin, J., "Suggested Practices for Registration of Internationalized Domain Names (IDN)", RFC 4290, DOI 10.17487/RFC4290, December 2005, https://www.rfc-editor.org/info/rfc4290.

[RFC5890] 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>

[RFC7940] Davies, K. and A. Freytag, "Representing Label Generation Rulesets Using XML", RFC 7940, DOI 10.17487/RFC7940, August 2016, https://www.rfc-editor.org/info/rfc7940.

[RFC8909] Lozano, G., "Registry Data Escrow Specification", RFC 8909, DOI 10.17487/RFC8909, November 2020, <https://www.rfc-editor.org/info/rfc8909>

[RZLGR] https://www.icann.org/resources/pages/root-zone-lgr-2015-06-21-en.

[STD69] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, August 2009, https://www.rfc-editor.org/info/std69.

[SubPro] https://gnso.icann.org/sites/default/files/file/field-file-attach/final-report-newgtld-subsequent-procedures-pdp-20jan21-en.pdf

[TANGO] https://www.knipp.de/it-es/tango?set-language=en

[UASG] https://uasg.tech

[UASG004] https://uasg.tech/download/uasg-004-use-cases-for-ua-readiness-evaluation-en/

[UASG004A] https://uasg.tech/download/uasg-004a-use-cases-for-ua-readiness-evaluation-data-en/

[UASG013] https://uasg.tech/download/uasg-013c-2-icann-case-study-en/

[UASG026] https://uasg.tech/download/uasg-026-ua-readiness-framework-en/

[UASG037A] https://uasg.tech/download/uasg-037a-ua-readiness-of-some-programming-language-libraries-and-frameworks-en/

[UASG043] UA-Ready Code Samples in Java, Python, and JavaScript, https://uasg.tech/download/uasg-043-ua-ready-code-samples-in-java-python-and-javascript-en/

[UASG-TECH] https://uasg.tech.