Network Working Group J. Klensin
Internet-Draft September 10, 2009
Intended status: Informational
Expires: March 14, 2010

Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale
draft-ietf-idnabis-rationale-12.txt

Status of this Memo

Copyright Notice

Abstract

Several years have passed since the original protocol for Internationalized Domain Names (IDNs) was completed and deployed. During that time, a number of issues have arisen, including the need to update the system to deal with newer versions of Unicode. Some of these issues require tuning of the existing protocols and the tables on which they depend. This document provides an overview of a revised system and provides explanatory material for its components.

Table of Contents

1. Introduction

1.1. Context and Overview

Internationalized Domain Names in Applications (IDNA) is a collection of standards that allow client applications to convert some Unicode mnemonics to an ASCII-compatible encoding form ("ACE") which is a valid DNS label containing only letters, digits, and hyphens. The specific form of ACE label used by IDNA is called an "A-label". A client can look up an exact A-label in the existing DNS, so A-labels do not require any extensions to DNS, upgrades of DNS servers or updates to low-level client libraries. An A-label is recognizable from the prefix "xn--" before the characters produced by the Punycode algorithm [RFC3492], thus a user application can identify an A-label and convert it into Unicode (or some local coded character set) for display.

On the registry side, IDNA allows a registry to offer Internationalized Domain Names (IDNs) for registration as A-labels. A registry may offer any subset of valid IDNs, and may apply any restrictions or bundling (grouping of similar labels together in one registration) appropriate for the context of that registry. Registration of labels is sometimes discussed separately from lookup, and is subject to a few specific requirements that do not apply to lookup.

DNS clients and registries are subject to some differences in requirements for handling IDNs. In particular, registries are urged to register only exact, valid A-labels, while clients might do some

mapping to get from otherwise-invalid user input to a valid A-label.

The first version of IDNA was published in 2003 and is referred to here as IDNA2003 to contrast it with the current version, which is known as IDNA2008 (after the year in which IETF work started on it). IDNA2003 consists of four documents: the IDNA base specification [RFC3490], Nameprep [RFC3491], Punycode [RFC3492], and Stringprep [RFC3454]. The current set of documents, IDNA2008, are not dependent on any of the IDNA2003 specifications other than the one for Punycode encoding. References to "these specifications" or "these documents" are to the entire IDNA2008 set listed in [IDNA2008-Defs]. The characters that are valid in A-labels are identified from rules listed in the Tables document [IDNA2008-Tables], but validity can be derived from the Unicode properties of those characters with a very few exceptions.

Traditionally, DNS labels are matched case-insensitively [RFC1034][RFC1035]. That convention was preserved in IDNA2003 by a case-folding operation that generally maps capital letters into

lower-case ones. However, if case rules are enforced from one language, another language sometimes loses the ability to treat two characters separately. Case-insensitivity is treated slightly differently in IDNA2008.

IDNA2003 used Unicode version 3.2 only. In order to keep up with new characters added in new versions of UNICODE, IDNA2008 decouples its rules from any particular version of UNICODE. Instead, the attributes of new characters in Unicode, supplemented by a small number of exception cases, determine how and whether the characters can be used in IDNA labels.

This document provides informational context for IDNA2008, including terminology, background, and policy discussions.

1.2. Discussion Forum

[[RFC Editor: please remove this section.]]

IDNA2008 is being discussed in the IETF "idnabis" Working Group and on the mailing list idna-update@alvestrand.no

1.3. Terminology

Terminology for IDNA2008 appears in [IDNA2008-Defs]. That document also contains a roadmap to the IDNA2008 document collection. No attempt should be made to understand this document without the definitions and concepts that appear there.

1.3.1. DNS "Name" Terminology

In the context of IDNs, the DNS term "name" has introduced some confusion as people speak of DNS labels in terms of the words or phrases of various natural languages. Historically, many of the "names" in the DNS have been mnemonics to identify some particular concept, object, or organization. They are typically rooted in some language because most people think in language-based ways. But, because they are mnemonics, they need not obey the orthographic conventions of any language: it is not a requirement that it be possible for them to be "words".

This distinction is important because the reasonable goal of an IDN effort is not to be able to write the great Klingon (or language of one's choice) novel in DNS labels but to be able to form a usefully broad range of mnemonics in ways that are as natural as possible in a very broad range of scripts.

1.3.2. New Terminology and Restrictions

These documents introduce new terminology, and precise definitions (in [IDNA2008-Defs]), for the terms "U-label", "A-Label", LDH-label (to which all valid pre-IDNA host names conformed), Reserved-LDH-label (R-LDH-label), XN-label, Fake-A-Label, and Non-Reserved-LDH-label (NR-LDH-label).

In addition, the term "putative label" has been adopted to refer to a label that may appear to meet certain definitional constraints but has not yet been sufficiently tested for validity.

These definitions are also illustrated in Figure 1 of the Definitions Document [IDNA2008-Defs]. R-LDH-labels contain "--" in the third and fourth character from the beginning of the label. In IDNA-aware

applications, only a subset of these reserved labels is permitted to be used, namely the A-label subset. A-labels are a subset of the R-LDH-labels that begin with the case-insensitive string "xn--". Labels that bear this prefix but which are not otherwise valid fall into the "Fake-A-label" category. The non-reserved labels (NR-LDH-labels) are implicitly valid since they do not trigger any resemblance to IDNA-landr NR-LDH-labels.

The creation of the Reserved-LDH category is required for three reasons:

- to prevent confusion with pre-IDNA coding forms;

- to permit future extensions that would require changing the prefix, no matter how unlikely those might be (see Section 7.4); and

- to reduce the opportunities for attacks via the Punycode encoding algorithm itself.

As with other documents in the IDNA2008 set, this document uses the term "registry" to describe any zone in the DNS. That term, and the terms "zone" or "zone administration", are interchangeable.

1.4. Objectives

These are the main objectives in revising IDNA.

- Use a more recent version of Unicode, and allow IDNA to be independent of Unicode versions, so that IDNA2008 need not be updated for implementations to adopt codepoints from new Unicode versions.

- Fix a very small number of code-point categorizations that have turned out to cause problems in the communities that use those code-points.

- Reduce the dependency on mapping, in order that the pre-mapped forms (which are not valid IDNA labels) tend to appear less often in various contexts, in favor of valid A-labels.

o Fix some details in the bidirectional codepoint handling algorithms.

1.5. Applicability and Function of IDNA

The IDNA specification solves the problem of extending the repertoire of characters that can be used in domain names to include a large subset of the Unicode repertoire.

IDNA does not extend DNS. Instead, the applications (and, by implication, the users) continue to see an exact-match lookup service. Either there is a single exactly-matching (subject to the base DNS requirement of case-insensitive ASCII matching) name or there is no match. This model has served the existing applications well, but it requires, with or without internationalized domain names, that users know the exact spelling of the domain names that are to be typed into applications such as web browsers and mail user agents. The introduction of the larger repertoire of characters potentially makes the set of misspellings larger, especially given that in some cases the same appearance, for example on a business card, might visually match several Unicode code points or several sequences of code points.

The IDNA standard does not require any applications to conform to it, nor does it retroactively change those applications. An application can elect to use IDNA in order to support IDN while maintaining interoperability with existing infrastructure. If an application wants to use non-ASCII characters in public DNS domain names, IDNA is the only currently-defined option. Adding IDNA support to an existing application entails changes to the application only, and leaves room for flexibility in front-end processing and more specifically in the user interface (see Section 6).

A great deal of the discussion of IDN solutions has focused on transition issues and how IDNs will work in a world where not all of the components have been updated. Proposals that were not chosen by the original IDN Working Group would have depended on updating of user applications, DNS resolvers, and DNS servers in order for a user to apply an internationalized domain name in any form or coding acceptable under that method. While processing must be performed

prior to or after access to the DNS, IDNA requires no changes to the

DNS protocol or any DNS servers or the resolvers on user's computers.

IDNA allows the graceful introduction of IDNs not only by avoiding upgrades to existing infrastructure (such as DNS servers and mail transport agents), but also by allowing some limited use of IDNs in applications by using the ASCII-encoded representation of the labels containing non-ASCII characters. While such names are user-unfriendly to read and type, and hence not optimal for user input, they can be used as a last resort to allow rudimentary IDN usage. For example, they might be the best choice for display if it were known that relevant fonts were not available on the user's computer. In order to allow user-friendly input and output of the IDNs and acceptance of some characters as equivalent to those to be processed according to the protocol, the applications need to be modified to conform to this specification.

This version of IDNA uses the Unicode character repertoire, for continuity with the original version of IDNA.

1.6. Comprehensibility of IDNA Mechanisms and Processing

One goal of IDNA2008, which is aided by the main goal of reducing the dependency on mapping, is to improve the general understanding of how IDNA works and what characters are permitted and what happens to them. Comprehensibility and predictability to users and registrants are important design goals for this effort. End-user applications have an important role to play in increasing this comprehensibility.

Any system that tries to handle international characters encounters some common problems. For example, a UI cannot display a character if no font for that character is available. In some cases, internationalization enables effective localization while maintaining some global uniformity but losing some universality.

It is difficult to even make suggestions for end-user applications to cope when characters and fonts are not available. Because display functions are rarely controlled by the types of applications that would call upon IDNA, such suggestions will rarely be very effective.

Converting between local character sets and normalized Unicode, if needed, is part of this set of user agent issues. This conversion introduces complexity in a system that is not Unicode-native. If a label is converted to a local character set that does not have all the needed characters, or that uses different character-coding principles, the user agent may have to add special logic to avoid or reduce loss of information.

The major difficulty may lie in accurately identifying the incoming character set and applying the correct conversion routine. Even more difficult, the local character coding system could be based on conceptually different assumptions than those used by Unicode (e.g., choice of font encodings used for publications in some Indic scripts). Those differences may not easily yield unambiguous conversions or interpretations even if each coding system is internally consistent and adequate to represent the local language and script.

IDNA2008 shifts responsibility for character mapping and other adjustments from the protocol (where it was located in IDNA2003) to pre-processing before invoking IDNA itself. The intent is that this change will lead to greater usage of fully-valid A-Labels or U-labels in display, transit and storage, which should aid comprehensibility and predictability. A careful look at pre-processing raises issues about what that pre-processing should do and at what point pre-processing becomes harmful, how universally consistent pre-processing algorithms can be, and how to be compatible with labels prepared in a IDNA2003 context. Those issues are discussed in Section 6 and in the separate document [IDNA2008-Mapping].

2. Processing in IDNA2008

These specifications separate Domain Name Registration and Lookup in the protocol specification. Although most steps in the two processes are similar, the separation reflects current practice in which per-registry (DNS zone) restrictions and special processing are applied at registration time but not during lookup. Another significant benefit is that separation facilitates incremental addition of permitted character groups to avoid freezing on one particular version of Unicode.

The actual registration and lookup protocols for IDNA2008 are specified in [IDNA2008-Protocol].

3. Permitted Characters: An Inclusion List

IDNA2008 adopts the inclusion model. A code-point is assumed to be invalid for IDN use unless it is included as part of a Unicode property-based rule or, in rare cases, included individually by an exception. When an implementation moves to a new version of Unicode, the rules may indicate new valid code-points.

This section provides an overview of the model used to establish the algorithm and character lists of [IDNA2008-Tables] and describes the

names and applicability of the categories used there. Note that the inclusion of a character in the first category group (Section 3.1.1) does not imply that it can be used indiscriminately; some characters are associated with contextual rules that must be applied as well.

The information given in this section is provided to make the rules, tables, and protocol easier to understand. The normative generating rules that correspond to this informal discussion appear in [IDNA2008-Tables] and the rules that actually determine what labels can be registered or looked up are in [IDNA2008-Protocol].

3.1. A Tiered Model of Permitted Characters and Labels

Moving to an inclusion model involves a new specification for the list of characters that are permitted in IDNs. In IDNA2003, character validity is independent of context and fixed forever (or until the standard is replaced). However, globally context-independent rules have proved to be impractical because some characters, especially those that are called "Join_Controls" in Unicode, are needed to make reasonable use of some scripts but have no visible effect in others. IDNA2003 prohibited those types of characters entirely by discarding them. We now have a consensus that under some conditions, these "joiner" characters are legitimately needed to allow useful mnemonics for some languages and scripts. In general, context-dependent rules help deal with characters (generally characters that would otherwise be prohibited entirely) that are used differently or perceived differently across different scripts, and allow the standard to be applied more appropriately in cases where a string is not universally handled the same way.

IDNA2008 divides all possible Unicode code-points into four categories: PROTOCOL-VALID, CONTEXTUAL RULE REQUIRED, DISALLOWED and UNASSIGNED.

3.1.1. PROTOCOL-VALID

Characters identified as "PROTOCOL-VALID" (often abbreviated "PVALID") are permitted in IDNs. Their use may be restricted by rules about the context in which they appear or by other rules that

apply to the entire label in which they are to be embedded. For example, any label that contains a character in this category that has a "right-to-left" property must be used in context with the "Bidi" rules (see [IDNA2008-Bidi]).

The term "PROTOCOL-VALID" is used to stress the fact that the presence of a character in this category does not imply that a given registry need accept registrations containing any of the characters in the category. Registries are still expected to apply judgment

about labels they will accept and to maintain rules consistent with those judgments (see [IDNA2008-Protocol] and Section 3.3).

Characters that are placed in the "PROTOCOL-VALID" category are expected to never be removed from it or reclassified. While theoretically characters could be removed from Unicode, such removal would be inconsistent with the Unicode stability principles (see [Unicode51], Appendix F) and hence should never occur.

3.1.2. CONTEXTUAL RULE REQUIRED

Some characters may be unsuitable for general use in IDNs but necessary for the plausible support of some scripts. The two most commonly-cited examples are the zero-width joiner and non-joiner characters (ZWJ, U+200D and ZWNJ, U+200C) but other characters may require special treatment because they would otherwise be DISALLOWED (typically because Unicode considers them punctuation or special symbols) but need to be permitted in limited contexts. Other characters are given this special treatment because they pose exceptional danger of being used to produce misleading labels or to cause unacceptable ambiguity in label matching and interpretation.

3.1.2.1. Contextual Restrictions

Characters with contextual restrictions are identified as "CONTEXTUAL RULE REQUIRED" and associated with a rule. The rule defines whether the character is valid in a particular string, and also whether the rule itself is to be applied on lookup as well as registration.

A distinction is made between characters that indicate or prohibit joining and ones similar to them (known as "CONTEXT-JOINER" or "CONTEXTJ") and other characters requiring contextual treatment

("CONTEXT-OTHER" or "CONTEXTO"). Only the former require full testing at lookup time.

It is important to note that these contextual rules cannot prevent all uses of the relevant characters that might be confusing or problematic. What they are expected do is to confine applicability of the characters to scripts (and narrower contexts) where zone administrators are knowledgeable enough about the use of those characters to be prepared to deal with them appropriately.

For example, a registry dealing with an Indic script that requires ZWJ and/or ZWNJ as part of the writing system is expected to understand where the characters have visible effect and where they do not and to make registration rules accordingly. By contrast, a registry dealing primarily with Latin or Cyrillic script might not be actively aware that the characters exist, much less about the

consequences of embedding them in labels drawn from those scripts and therefore should avoid accepting registrations containing those characters, at least in Latin or Cyrillic-script labels.

3.1.2.2. Rules and Their Application

Rules have descriptions such as "Must follow a character from Script XYZ", "Must occur only if the entire label is in Script ABC", or "Must occur only if the previous and subsequent characters have the DFG property". The actual rules may be DEFINED or NULL. If present, they may have values of "True" (character may be used in any position in any label), "False" (character may not be used in any label), or may be a set of procedural rules that specify the context in which the character is permitted.

Examples of descriptions of typical rules, stated informally and in English, include "Must follow a character from Script XYZ", "Must occur only if the entire label is in Script ABC", "Must occur only if the previous and subsequent characters have the DFG property".

Because it is easier to identify these characters than to know that they are actually needed in IDNs or how to establish exactly the right rules for each one, a rule may have a null value in a given version of the tables. Characters associated with null rules are not permitted to appear in putative labels for either registration or

lookup. Of course, a later version of the tables might contain a non-null rule.

The actual rules and their descriptions are in [IDNA2008-Tables]. [[anchor9: ??? Section number would be good here.]] That document also specifies the creation of a registry for future rules.

3.1.3. DISALLOWED

Some characters are inappropriate for use in IDNs and are thus excluded for both registration and lookup (i.e., IDNA-conforming applications performing name lookup should verify that these characters are absent; if they are present, the label strings should be rejected rather than converted to A-labels and looked up. Some of these characters are problematic for use in IDNs (such as the FRACTION SLASH character, U+2044), while some of them (such as the various HEART symbols, e.g., U+2665, U+2661, and U+2765, see Section 7.6) simply fall outside the conventions for typical identifiers (basically letters and numbers).

Of course, this category would include code points that had been removed entirely from Unicode should such removals ever occur.

Characters that are placed in the "DISALLOWED" category are expected to never be removed from it or reclassified. If a character is classified as "DISALLOWED" in error and the error is sufficiently problematic, the only recourse would be either to introduce a new code point into Unicode and classify it as "PROTOCOL-VALID" or for the IETF to accept the considerable costs of an incompatible change and replace the relevant RFC with one containing appropriate exceptions.

There is provision for exception cases but, in general, characters are placed into "DISALLOWED" if they fall into one or more of the following groups:

- o The character is a compatibility equivalent for another character. In slightly more precise Unicode terms, application of normalization method NFKC to the character yields some other character.

o The character is an upper-case form or some other form that is mapped to another character by Unicode casefolding.

o The character is a symbol or punctuation form or, more generally, something that is not a letter, digit, or a mark that is used to form a letter or digit.

3.1.4. UNASSIGNED

For convenience in processing and table-building, code points that do not have assigned values in a given version of Unicode are treated as belonging to a special UNASSIGNED category. Such code points are prohibited in labels to be registered or looked up. The category differs from DISALLOWED in that code points are moved out of it by the simple expedient of being assigned in a later version of Unicode (at which point, they are classified into one of the other categories as appropriate).

The rationale for restricting the processing of UNASSIGNED characters is simply that the properties of such code points cannot be completely known until actual characters are assigned to them. If, for example, such a code point was permitted to be included in a label to be looked up, and the code point was later to be assigned to a character that required some set of contextual rules, un-updated instances of IDNA-aware software might permit lookup of labels containing the previously-unassigned characters while updated versions of IDNA-aware software might restrict their use in lookup, depending on the contextual rules. It should be clear that under no circumstance should an UNASSIGNED character be permitted in a label to be registered as part of a domain name.

3.2. Registration Policy

While these recommendations cannot and should not define registry policies, registries should develop and apply additional restrictions as needed to reduce confusion and other problems. For example, it is generally believed that labels containing characters from more than one script are a bad practice although there may be some important exceptions to that principle. Some registries may choose to restrict registrations to characters drawn from a very small number of scripts. For many scripts, the use of variant techniques such as those as described in RFC 3843 [RFC3743] and RFC 4290 [RFC4290], and

illustrated for Chinese by the tables described in RFC 4713 [RFC4713] may be helpful in reducing problems that might be perceived by users.

In general, users will benefit if registries only permit characters from scripts that are well-understood by the registry or its advisers. If a registry decides to reduce opportunities for confusion by constructing policies that disallow characters used in historic writing systems or characters whose use is restricted to specialized, highly technical contexts, some relevant information may be found in Section 2.4 "Specific Character Adjustments", Table 4 "Candidate Characters for Exclusion from Identifiers" of [Unicode-UAX31] and Section 3.1. "General Security Profile for Identifiers" in [Unicode-Security].

The requirement (in [IDNA2008-Protocol] [[anchor10: ?? Section number]]) that registration procedures use only U-labels and/or A-labels is intended to ensure that registrants are fully aware of exactly what is being registered as well as encouraging use of those canonical forms. That provision should not be interpreted as requiring that registrant need to provide characters in a particular code sequence. Registrant input conventions and management are part of registrant-registrar interactions and relationships between registries and registrars and are outside the scope of these standards.

It is worth stressing that these principles of policy development and application apply at all levels of the DNS, not only, e.g., TLD or SLD registrations and that even a trivial, "anything is permitted that is valid under the protocol" policy is helpful in that it helps users and application developers know what to expect.

3.3. Layered Restrictions: Tables, Context, Registration, Applications

The character rules in IDNA2008 are based on the realization that there is no single magic bullet for any of the security, confusability, or other issues associated with IDNs. Instead, the specifications define a variety of approaches. The character tables

are the first mechanism, protocol rules about how those characters are applied or restricted in context are the second, and those two in combination constitute the limits of what can be done in the protocol. As discussed in the previous section (Section 3.2),

registries are expected to restrict what they permit to be registered, devising and using rules that are designed to optimize the balance between confusion and risk on the one hand and maximum expressiveness in mnemonics on the other.

In addition, there is an important role for user agents in warning against label forms that appear problematic given their knowledge of local contexts and conventions. Of course, no approach based on naming or identifiers alone can protect against all threats.

4. Issues that Constrain Possible Solutions

4.1. Display and Network Order

Domain names are always transmitted in network order (the order in which the code points are sent in protocols), but may have a different display order (the order in which the code points are displayed on a screen or paper). When a domain name contains characters that are normally written right to left, display order may be affected although network order is not. It gets even more complicated if left to right and right to left labels are adjacent to each other within a domain name. The decision about the display order is ultimately under the control of user agents --including Web browsers, mail clients, hosted Web applications and many more -- which may be highly localized. Should a domain name abc.def, in which both labels are represented in scripts that are written right to left, be displayed as fed.cba or cba.fed? Applications that are in deployment today are already diverse, and one can find examples of either choice.

The picture changes once again when an IDN appears in a Internationalized Resource Identifier (IRI) [RFC3987]. An IRI or Internationalized Email address contains elements other than the domain name. For example, IRIs contain protocol identifiers and field delimiter syntax such as "http://" or "mailto:" while email addresses contain the "@" to separate local parts from domain names. An IRI in network order begins with "http://" followed by domain labels in network order, thus "http://abc.def".

User agents are not required to display and allow input of IRIs directly but often do so. Implementors have to choose whether the overall direction of these strings will always be left to right (or right to left) for an IRI or email address. The natural order for a

user typing a domain name on a right to left system is fed.cba. Should the R2L user agent reverse the entire domain name each time a domain name is typed? Does this change if the user types "http://" right before typing a domain name, thus implying that the user is beginning at the beginning of the network order IRI? Experience in the 1980s and 1990s with mixing systems in which domain name labels were read in network order (left to right) and those in which those labels were read right to left would predict a great deal of confusion.

If each implementation of each application makes its own decisions on these issues, users will develop heuristics that will sometimes fail when switching applications. However, while some display order conventions, voluntarily adopted, would be desirable to reduce confusion, such suggestions are beyond the scope of these specifications.

4.2. Entry and Display in Applications

Applications can accept and display domain names using any character set or character coding system. The IDNA protocol does not necessarily affect the interface between users and applications. An IDNA-aware application can accept and display internationalized domain names in two formats: the internationalized character set(s) supported by the application (i.e., an appropriate local representation of a U-label), and as an A-label. Applications may allow the display of A-labels, but are encouraged to not do so except as an interface for special purposes, possibly for debugging, or to cope with display limitations. In general, they should allow, but not encourage, user input of A-labels. A-labels are opaque, ugly, and malicious variations on them are not easily detected by users. Where possible, they should thus only be exposed when they are absolutely needed. Because IDN labels can be rendered either as A-labels or U-labels, the application may reasonably have an option for the user to select the preferred method of display. Rendering the U-label should normally be the default.

Domain names are often stored and transported in many places. For example, they are part of documents such as mail messages and web pages. They are transported in many parts of many protocols, such as both the control commands of SMTP and associated message body parts, and in the headers and the body content in HTTP. It is important to remember that domain names appear both in domain name slots and in the content that is passed over protocols.

In protocols and document formats that define how to handle specification or negotiation of charsets, labels can be encoded in

any charset allowed by the protocol or document format. If a

protocol or document format only allows one charset, the labels must be given in that charset. Of course, not all charsets can properly represent all labels. If a U-label cannot be displayed in its entirety, the only choice (without loss of information) may be to display the A-label.

Where a protocol or document format allows IDNs, labels should be in whatever character encoding and escape mechanism the protocol or document format uses at that place. This provision is intended to prevent situations in which, e.g., UTF-8 domain names appear embedded in text that is otherwise in some other character coding.

All protocols that use domain name slots (See Section 2.3.1.6 in [IDNA2008-Defs]) already have the capacity for handling domain names in the ASCII charset. Thus, A-labels can inherently be handled by those protocols.

These documents do not specify required mappings between one character or code point and others. An extended discussion of mapping issues occurs in Section 6 and specific recommendations appear in [IDNA2008-Mapping]. In general, IDNA2008 prohibits characters that would be mapped to others by normalization or other rules. As examples, while mathematical characters based on Latin ones are accepted as input to IDNA2003, they are prohibited in IDNA2008. Similarly, upper-case characters, double-width characters, and other variations are prohibited as IDNA input although mapping them as needed in user interfaces is strongly encouraged.

Since the rules in [IDNA2008-Tables] have the effect that only strings that are not transformed by NFKC are valid, if an application chooses to perform NFKC normalization before lookup, that operation is safe since this will never make the application unable to look up any valid string. However, as discussed above, the application cannot guarantee that any other application will perform that mapping, so it should be used only with caution and for informed users.

In many cases these prohibitions should have no effect on what the user can type as input to the lookup process. It is perfectly reasonable for systems that support user interfaces to perform some

character mapping that is appropriate to the local environment. This would normally be done prior to actual invocation of IDNA. At least conceptually, the mapping would be part of the Unicode conversions discussed above and in [IDNA2008-Protocol]. However, those changes will be local ones only -- local to environments in which users will clearly understand that the character forms are equivalent. For use in interchange among systems, it appears to be much more important that U-labels and A-labels can be mapped back and forth without loss

of information.

One specific, and very important, instance of this strategy arises with case-folding. In the ASCII-only DNS, names are looked up and matched in a case-independent way, but no actual case-folding occurs. Names can be placed in the DNS in either upper or lower case form (or any mixture of them) and that form is preserved, returned in queries, and so on. IDNA2003 approximated that behavior for non-ASCII strings by performing case-folding at registration time (resulting in only lower-case IDNs in the DNS) and when names were looked up.

As suggested earlier in this section, it appears to be desirable to do as little character mapping as possible as long as Unicode works correctly (e.g., NFC mapping to resolve different codings for the same character is still necessary although the specifications require that it be performed prior to invoking the protocol) in order to make the mapping between A-labels and U-labels idempotent. Case-mapping is not an exception to this principle. If only lower case characters can be registered in the DNS (i.e., be present in a U-label), then IDNA2008 should prohibit upper-case characters as input even though user interfaces to applications should probably map those characters. Some other considerations reinforce this conclusion. For example, in ASCII case-mapping for individual characters, uppercase(character) must be equal to uppercase(lowercase(character)). That may not be true with IDNs. In some scripts that use case distinctions, there are a few characters that do not have counterparts in one case or the other. The relationship between upper case and lower case may even be language-dependent, with different languages (or even the same language in different areas) expecting different mappings. User agents can meet the expectations of users who are accustomed to the case-insensitive DNS environment by performing case folding prior to IDNA processing, but the IDNA procedures themselves should neither require such mapping nor expect them when they are not natural to the

localized environment.

4.3. Linguistic Expectations: Ligatures, Digraphs, and Alternate Character Forms

Users have expectations about character matching or equivalence that are based on their own languages and the orthography of those languages. These expectations may not always be met in a global system, especially if multiple languages are written using the same script but using different conventions. Some examples:

- A Norwegian user might expect a label with the ae-ligature to be treated as the same label as one using the Swedish spelling with a-diaeresis even though applying that mapping to English would be astonishing to users.

- A user in German might expect a label with an o-umlaut and a label that had "oe" substituted, but was otherwise the same, treated as equivalent even though that substitution would be a clear error in Swedish.

- A Chinese user might expect automatic matching of Simplified and Traditional Chinese characters, but applying that matching for Korean or Japanese text would create considerable confusion.

- An English user might expect "theater" and "theatre" to match.

A number of languages use alphabetic scripts in which single phonemes are written using two characters, termed a "digraph", for example, the "ph" in "pharmacy" and "telephone". (Such characters can also appear consecutively without forming a digraph, as in "tophat".) Certain digraphs may be indicated typographically by setting the two characters closer together than they would be if used consecutively to represent different phonemes. Some digraphs are fully joined as ligatures. For example, the word "encyclopaedia" is sometimes set with a U+00E6 LATIN SMALL LIGATURE AE. When ligature and digraph forms have the same interpretation across all languages that use a given script, application of Unicode normalization generally resolves the differences and causes them to match. When they have different interpretations, matching must utilize other methods, presumably chosen at the registry level, or users must be educated to understand that matching will not occur.

The nature of the problem can be illustrated by many words in the Norwegian language, where the "ae" ligature is the 27th letter of a 29-letter extended Latin alphabet. It is equivalent to the 28th letter of the Swedish alphabet (also containing 29 letters), U+00E4 LATIN SMALL LETTER A WITH DIAERESIS, for which an "ae" cannot be substituted according to current orthographic standards. That character (U+00E4) is also part of the German alphabet where, unlike in the Nordic languages, the two-character sequence "ae" is usually treated as a fully acceptable alternate orthography for the "umlauted a" character. The inverse is however not true, and those two characters cannot necessarily be combined into an "umlauted a". This also applies to another German character, the "umlauted o" (U+00F6 LATIN SMALL LETTER O WITH DIAERESIS) which, for example, cannot be used for writing the name of the author "Goethe". It is also a letter in the Swedish alphabet where, like the "a with diaeresis", it cannot be correctly represented as "oe" and in the Norwegian alphabet, where it is represented, not as "o with diaeresis", but as "slashed o", U+00F8.

Some of the ligatures that have explicit code points in Unicode were given special handling in IDNA2003 and now pose additional problems

in transition. See Section 7.2.

Additional cases with alphabets written right to left are described in Section 4.5.

Matching and comparison algorithm selection often requires information about the language being used, context, or both -- information that is not available to IDNA or the DNS. Consequently, these specifications make no attempt to treat combined characters in any special way. A registry that is aware of the language context in which labels are to be registered, and where that language sometimes (or always) treats the two- character sequences as equivalent to the combined form, should give serious consideration to applying a "variant" model [RFC3743][RFC4290], or to prohibiting registration of one of the forms entirely, to reduce the opportunities for user confusion and fraud that would result from the related strings being registered to different parties.

4.4. Case Mapping and Related Issues

In the DNS, ASCII letters are stored with their case preserved. Matching during the query process is case-independent, but none of the information that might be represented by choices of case has been lost. That model has been accidentally helpful because, as people have created DNS labels by catenating words (or parts of words) to form labels, case has often been used to distinguish among components and make the labels more memorable.

Since DNS servers do not get involved in parsing IDNs, they cannot do case-independent matching. Thus, keeping the cases separate in lookup or registration, and doing matching at the server, is not feasible with IDNA or any similar approach. Case-matching must be done, if desired, by IDN clients even though it wasn't done by ASCII-only DNS clients. That situation was recognized in IDNA2003 and nothing in these specifications fundamentally changes it or could do so. In IDNA2003, all characters are case-folded and mapped by clients in a standardized step.

Some characters do not have upper case forms. For example the Unicode case folding operation maps Greek Final Form Sigma (U+03C2) to the medial form (U+03C3) and maps Eszett (German Sharp S, U+00DF) to "ss". Neither of these mappings is reversible because the upper case of U+03C3 is the Upper Case Sigma (U+03A3) and "ss" is an ASCII string. IDNA2008 permits, at the risk of some incompatibility, slightly more flexibility in this area by avoiding case folding and treating these characters as themselves. Approaches to handling one-way mappings are discussed in Section 7.2.

Because IDNA2003 maps Final Sigma and Eszett to other characters, and the reverse mapping is never possible, that in some sense means that neither Final Sigma nor Eszett can be represented in a IDNA2003 IDN. With IDNA2008, both characters can be used in an IDN and so the A-label used for lookup for any U-label containing those characters, is now different. See Section 7.1 for a discussion of what kinds of changes might require the IDNA prefix to change; after extended discussions, the WG came to consensus that the change for these characters did not justify a prefix change.

4.5. Right to Left Text

In order to be sure that the directionality of right to left text is unambiguous, IDNA2003 required that any label in which right to left characters appear both starts and ends with them and that it not include any characters with strong left to right properties (that excludes other alphabetic characters but permits European digits). Any other string that contains a right to left character and does not meet those requirements is rejected. This is one of the few places where the IDNA algorithms (both in IDNA2003 and in IDAN2008) examine an entire label, not just individual characters. The algorithmic model used in IDNA2003 rejects the label when the final character in a right to left string requires a combining mark in order to be correctly represented.

That prohibition is not acceptable for writing systems for languages written with consonantal alphabets to which diacritical vocalic systems are applied, and for languages with orthographies derived from them where the combining marks may have different functionality. In both cases the combining marks can be essential components of the orthography. Examples of this are Yiddish, written with an extended Hebrew script, and Dhivehi (the official language of Maldives) which is written in the Thaana script (which is, in turn, derived from the Arabic script). IDNA2008 removes the restriction on final combining characters with a new set of rules for right to left scripts and their characters. Those new rules are specified in [IDNA2008-Bidi].

5. IDNs and the Robustness Principle

The "Robustness Principle" is often stated as "Be conservative about what you send and liberal in what you accept" (See, e.g., Section 1.2.2 of the applications-layer Host Requirements specification [RFC1123]) This principle applies to IDNA. In applying the principle to registries as the source ("sender") of all registered and useful IDNs, registries are responsible for being conservative about what they register and put out in the Internet. For IDNs to work well, zone administrators (registries) must have and require sensible

policies about what is registered -- conservative policies -- and implement and enforce them.

Conversely, lookup applications are expected to reject labels that clearly violate global (protocol) rules (no one has ever seriously

claimed that being liberal in what is accepted requires being stupid). However, once one gets past such global rules and deals with anything sensitive to script or locale, it is necessary to assume that garbage has not been placed into the DNS, i.e., one must be liberal about what one is willing to look up in the DNS rather than guessing about whether it should have been permitted to be registered.

If a string cannot be successfully found in the DNS after the lookup processing described here, it makes no difference whether it simply wasn't registered or was prohibited by some rule at the registry. Application implementors should be aware that where DNS wildcards are used, the ability to successfully resolve a name does not guarantee that it was actually registered.

6. Front-end and User Interface Processing for Lookup

Domain names may be identified and processed in many contexts. They may be typed in by users either by themselves or embedded in an identifier such as email addresses, URIs, or IRIs. They may occur in running text or be processed by one system after being provided in another. Systems may try to normalize URLs to determine (or guess) whether a reference is valid or two references point to the same object without actually looking the objects up (comparison without lookup is necessary for URI types that are not intended to be resolved). Some of these goals may be more easily and reliably satisfied than others. While there are strong arguments for any domain name that is placed "on the wire" -- transmitted between systems -- to be in the zero-ambiguity forms of A-labels, it is inevitable that programs that process domain names will encounter U-labels or variant forms.

An application that implements the IDNA protocol [IDNA2008-Protocol] will always take any user input and convert it to a set of Unicode code points. That user input may be acquired by any of several different input methods, all with differing conversion processes to be taken into consideration (e.g., typed on a keyboard, written by hand onto some sort of digitizer, spoken into a microphone and interpreted by a speech-to-text engine, etc.). The process of taking any particular user input and mapping it into a Unicode code point may be a simple one: If a user strikes the "A" key on a US English keyboard, without any modifiers such as the "Shift" key held down, in

order to draw a Latin small letter A ("a"), many (perhaps most) modern operating system input methods will produce to the calling application the code point U+0061, encoded in a single octet.

Sometimes the process is somewhat more complicated: a user might strike a particular set of keys to represent a combining macron followed by striking the "A" key in order to draw a Latin small letter A with a macron above it. Depending on the operating system, the input method chosen by the user, and even the parameters with which the application communicates with the input method, the result might be the code point U+0101 (encoded as two octets in UTF-8 or UTF-16, four octets in UTF-32, etc.), the code point U+0061 followed by the code point U+0304 (again, encoded in three or more octets, depending upon the encoding used) or even the code point U+FF41 followed by the code point U+0304 (and encoded in some form). And these examples leave aside the issue of operating systems and input methods that do not use Unicode code points for their character set.

In every case, applications (with the help of the operating systems on which they run and the input methods used) need to perform a mapping from user input into Unicode code points.

The original version of the IDNA protocol [RFC3490] used a model whereby input was taken from the user, mapped (via whatever input method mechanisms were used) to a set of Unicode code points, and then further mapped to a set of Unicode code points using the Nameprep profile specified in [RFC3491]. In this procedure, there are two separate mapping steps: First, a mapping done by the input method (which might be controlled by the operating system, the application, or some combination) and then a second mapping performed by the Nameprep portion of the IDNA protocol. The mapping done in Nameprep includes a particular mapping table to re-map some characters to other characters, a particular normalization, and a set of prohibited characters.

Note that the result of the two step mapping process means that the mapping chosen by the operating system or application in the first step might differ significantly from the mapping supplied by the Nameprep profile in the second step. This has advantages and disadvantages. Of course, the second mapping regularizes what gets looked up in the DNS, making for better interoperability between implementations which use the Nameprep mapping. However, the application or operating system may choose mappings in their input methods, which when passed through the second (Nameprep) mapping result in characters that are "surprising" to the end user.

The other important feature of the original version of the IDNA protocol is that, with very few exceptions, it assumes that any set

of Unicode code points provided to the Nameprep mapping can be mapped into a string of Unicode code points that are "sensible", even if that means mapping some code points to nothing (that is, removing the code points from the string). This allowed maximum flexibility in input strings.

The present version of IDNA differs significantly in approach from the original version. First and foremost, it does not provide explicit mapping instructions. Instead, it assumes that the application (perhaps via an operating system input method) will do whatever mapping it requires to convert input into Unicode code points. This has the advantage of giving flexibility to the application to choose a mapping that is suitable for its user given specific user requirements, and avoids the two-step mapping of the original protocol. Instead of a mapping, the current version of IDNA provides a set of categories that can be used to specify the valid code points allowed in a domain name.

In principle, an application ought to take user input of a domain name and convert it to the set of Unicode code points that represent the domain name the user intends. As a practical matter, of course, determining user intent is a tricky business, so an application needs to choose a reasonable mapping from user input. That may differ based on the particular circumstances of a user, depending on locale, language, type of input method, etc. It is up to the application to make a reasonable choice.

7. Migration from IDNA2003 and Unicode Version Synchronization

7.1. Design Criteria

As mentioned above and in RFC 4690, two key goals of the IDNA2008 design are

o to enable applications to be agnostic about whether they are being run in environments supporting any Unicode version from 3.2 onward,

o to permit incrementally adding new characters, character groups, scripts, and other character collections as they are incorporated into Unicode, doing so without disruption and, in the long term,

without "heavy" processes (an IETF consensus process is required by the IDNA2008 specifications and is expected to be required and used until significant experience accumulates with IDNA operations and new versions of Unicode).

7.1.1. Summary and Discussion of IDNA Validity Criteria

The general criteria for a label to be considered valid under IDNA are (the actual rules are rigorously defined in the "Protocol" and "Tables" documents):

- o The characters are "letters", marks needed to form letters, numerals, or other code points used to write words in some language. Symbols, drawing characters, and various notational characters are intended to be permanently excluded. There is no evidence that they are important enough to Internet operations or internationalization to justify expansion of domain names beyond the general principle of "letters, digits, and hyphen". (Additional discussion and rationale for the symbol decision appears in Section 7.6).

- o Other than in very exceptional cases, e.g., where they are needed to write substantially any word of a given language, punctuation characters are excluded. The fact that a word exists is not proof that it should be usable in a DNS label and DNS labels are not expected to be usable for multiple-word phrases (although they are certainly not prohibited if the conventions and orthography of a particular language cause that to be possible).

- o Characters that are unassigned (have no character assignment at all) in the version of Unicode being used by the registry or application are not permitted, even on lookup. The issues involved in this decision are discussed in Section 7.7.

- o Any character that is mapped to another character by a current version of NFKC is prohibited as input to IDNA (for either registration or lookup). With a few exceptions, this principle excludes any character mapped to another by Nameprep [RFC3491].

The principles above drive the design of rules that are specified

exactly in [IDNA2008-Tables]. Those rules identify the characters that are valid under IDNA. The rules themselves are normative, and the tables are derived from them, rather than vice versa.

7.1.2. Labels in Registration

Any label registered in a DNS zone must be validated -- i.e., the criteria for that label must be met -- in order for applications to work as intended. This principle is not new. For example, since the DNS was first deployed, zone administrators have been expected to verify that names meet "hostname" requirements [RFC0952] where those requirements are imposed by the expected applications. Other applications contexts, such as the later addition of special service

location formats [RFC2782] imposed new requirements on zone administrators. For zones that will contain IDNs, support for Unicode version-independence requires restrictions on all strings placed in the zone. In particular, for such zones:

- Any label that appears to be an A-label, i.e., any label that starts in "xn--", must be valid under IDNA, i.e., they must be valid A-labels, as discussed in Section 2 above.

- The Unicode tables (i.e., tables of code points, character classes, and properties) and IDNA tables (i.e., tables of contextual rules such as those that appear in the Tables document), must be consistent on the systems performing or validating labels to be registered. Note that this does not require that tables reflect the latest version of Unicode, only that all tables used on a given system are consistent with each other.

Under this model, registry tables will need to be updated (both the Unicode-associated tables and the tables of permitted IDN characters) to enable a new script or other set of new characters. The registry will not be affected by newer versions of Unicode, or newly-authorized characters, until and unless it wishes to support them. The zone administrator is responsible for verifying validity for IDNA as well as its local policies -- a more extensive set of checks than are required for looking up the labels. Systems looking up or resolving DNS labels, especially IDN DNS labels, must be able to assume that applicable registration rules were followed for names

entered into the DNS.

7.1.3. Labels in Lookup

Anyone looking up a label in a DNS zone is required to

- o Maintain IDNA and Unicode tables that are consistent with regard to versions, i.e., unless the application actually executes the classification rules in [IDNA2008-Tables], its IDNA tables must be derived from the version of Unicode that is supported more generally on the system. As with registration, the tables need not reflect the latest version of Unicode but they must be consistent.

- o Validate the characters in labels to be looked up only to the extent of determining that the U-label does not contain "DISALLOWED" code points or code points that are unassigned in its version of Unicode.

- o Validate the label itself for conformance with a small number of whole-label rules. In particular, it must verify that

  * there are no leading combining marks,

  * the "bidi" conditions are met if right to left characters appear,

  * any required contextual rules are available, and

  * any contextual rules that are associated with Joiner Controls (and "CONTEXTJ" characters more generally) are tested.

- o Do not reject labels based on other contextual rules about characters, including mixed-script label prohibitions. Such rules may be used to influence presentation decisions in the user interface, but not to avoid looking up domain names.

To further clarify the rules about handling characters that require contextual rules, note that one can have a context-required character (i.e., one that requires a rule), but no rule. In that case, the

character is treated the same way DISALLOWED characters are treated, until and unless a rule is supplied. That state is more or less equivalent to "the idea of permitting this character is accepted in principle, but it won't be permitted in practice until consensus is reached on a safe way to use it".

The ability to add a rule more or less exempts these characters from the prohibition against reclassifying characters from DISALLOWED to PVALID.

And, obviously, "no rule" is different from "have a rule, but the test either succeeds or fails".

Lookup applications that follow these rules, rather than having their own criteria for rejecting lookup attempts, are not sensitive to version incompatibilities with the particular zone registry associated with the domain name except for labels containing characters recently added to Unicode.

An application or client that processes names according to this protocol and then resolves them in the DNS will be able to locate any name that is registered, as long as those registrations are valid under IDNA and its version of the IDNA tables is sufficiently up-to-date to interpret all of the characters in the label. Messages to users should distinguish between "label contains an unallocated code point" and other types of lookup failures. A failure on the basis of an old version of Unicode may lead the user to a desire to upgrade to

a newer version, but will have no other ill effects (this is consistent with behavior in the transition to the DNS when some hosts could not yet handle some forms of names or record types).

7.2. Changes in Character Interpretations

In those scripts that make case distinctions, there are a few characters for which an obvious and unique upper case character has not historically been available to match a lower case one or vice versa. For those characters, the mappings used in constructing the Stringprep tables for IDNA2003, performed using the Unicode CaseFold operation (See Section 5.8 of the Unicode Standard [Unicode51]), generate different characters or sets of characters. Those operations are not reversible and lose even more information than

traditional upper case or lower case transformations, but are more useful than those transformations for comparison purposes. Two notable characters of this type are the German character Eszett (Sharp S, U+00DF) and the Greek Final Form Sigma (U+03C2). The former is case-folded to the ASCII string "ss", the latter to a medial (Lower Case) Sigma (U+03C3).

The decision to eliminate mandatory and standardized mappings, including case folding, from the IDNA2008 protocol in order to make A-labels and U-labels idempotent made these characters problematic. If they were to be disallowed, important words and mnemonics could not be written in orthographically reasonable ways. If they were to be permitted as distinct characters, there would be no information loss and registries would have more flexibility, but IDNA2003 and IDNA2008 lookups might result in different A-labels.

With the understanding that there would be incompatibility either way but a judgment that the incompatibility was not significant enough to justify a prefix change, the WG concluded that Eszett and Final Form Sigma should be treated as distinct and Protocol-Valid characters.

Registries, especially those maintaining zones for third parties, must decide how to introduce a new service in a way that does not create confusion or significantly weaken or invalidate existing identifiers. This is not a new problem; registries were faced with similar issues when IDNs were introduced and when other new forms of strings have been permitted as labels.

There are several approaches to problems of this type. Without any preference or claim to completeness, some of these, all of which have been used by registries in the past for similar transitions, are:

o Do not permit use of the newly-available character at the registry level. This might cause lookup failures if a domain name were to be written with the expectation of the IDNA2003 mapping behavior, but would eliminate any possibility of false matches.

o Hold a "sunrise"-like arrangement in which holders of labels containing "ss" in the Eszett case or Lower Case Sigma are given

priority (and perhaps other benefits) for registering the corresponding string containing Eszett or Final Sigma respectively.

- o Adopt some sort of "variant" approach in which registrants obtain labels with both character forms.

- o Adopt a different form of "variant" approach in which registration of additional names is either not permitted at all or permitted only by the registrant who already has one of the names.

7.3. Character Mapping

As discussed at length in Section 6, IDNA2003, via Nameprep (see Section 7.5), mapped many characters into related ones. Those mappings no longer exist as requirements in IDNA2008. These specifications strongly prefer that only A-labels or U-labels be used in protocol contexts and as much as practical more generally. IDNA2008 does anticipate situations in which some mapping at the time of user input into lookup applications is appropriate and desirable. The issues are discussed in Section 6 and specific recommendations are made in [IDNA2008-Mapping].

7.4. The Question of Prefix Changes

The conditions that would require a change in the IDNA ACE prefix ("xn--" for the version of IDNA specified in [RFC3490]) have been a great concern to the community. A prefix change would clearly be necessary if the algorithms were modified in a manner that would create serious ambiguities during subsequent transition in registrations. This section summarizes our conclusions about the conditions under which changes in prefix would be necessary and the implications of such a change.

7.4.1. Conditions Requiring a Prefix Change

An IDN prefix change is needed if a given string would be looked up or otherwise interpreted differently depending on the version of the protocol or tables being used. An IDNA upgrade would require a prefix change if, and only if, one of the following four conditions were met:

1. The conversion of an A-label to Unicode (i.e., a U-label) yields one string under IDNA2003 (RFC3490) and a different string under IDNA2008.

2. In a significant number of cases, an input string that is valid under IDNA2003 and also valid under IDNA2008 yields two different A-labels with the different versions. This condition is believed to be essentially equivalent to the one above except for a very small number of edge cases which may not justify a prefix change (See Section 7.2).

   Note that if the input string is valid under one version and not valid under the other, this condition does not apply. See the first item in Section 7.4.2, below.

3. A fundamental change is made to the semantics of the string that is inserted in the DNS, e.g., if a decision were made to try to include language or script information in the encoding in addition to the string itself.

4. A sufficiently large number of characters is added to Unicode so that the Punycode mechanism for block offsets can no longer reference the higher-numbered planes and blocks. This condition is unlikely even in the long term and certain not to arise in the next several years.

7.4.2. Conditions Not Requiring a Prefix Change

As a result of the principles described above, none of the following changes require a new prefix:

1. Prohibition of some characters as input to IDNA. This may make names that are now registered inaccessible, but does not change those names.

2. Adjustments in IDNA tables or actions, including normalization definitions, that affect characters that were already invalid under IDNA2003.

3. Changes in the style of the IDNA definition that does not alter the actions performed by IDNA.

7.4.3. Implications of Prefix Changes

While it might be possible to make a prefix change, the costs of such a change are considerable. Registries could not convert all IDNA2003 ("xn--") registrations to a new form at the same time and synchronize that change with applications supporting lookup. Unless all existing

registrations were simply to be declared invalid (and perhaps even then) systems that needed to support both labels with old prefixes and labels with new ones would first process a putative label under the IDNA2008 rules and try to look it up and then, if it were not found, would process the label under IDNA2003 rules and look it up again. That process could significantly slow down all processing that involved IDNs in the DNS especially since a fully-qualified name might contain a mixture of labels that were registered with the old and new prefixes. That would make DNS caching very difficult. In addition, looking up the same input string as two separate A-labels creates some potential for confusion and attacks, since the labels could map to different targets and then resolve to different entries in the DNS.

Consequently, a prefix change is to be avoided if at all possible, even if it means accepting some IDNA2003 decisions about character distinctions as irreversible and/or giving special treatment to edge cases.

7.5. Stringprep Changes and Compatibility

The Nameprep [RFC3491] specification, a key part of IDNA2003, is a profile of Stringprep [RFC3454]. While Nameprep is a Stringprep profile specific to IDNA, Stringprep is used by a number of other protocols. Were Stringprep to be modified by IDNA2008, those changes to improve the handling of IDNs could cause problems for non-DNS uses, most notably if they affected identification and authentication protocols. Several elements of IDNA2008 give interpretations to strings prohibited under IDNA2003 or prohibit strings that IDNA2003 permitted. Those elements include the proposed new inclusion tables [IDNA2008-Tables], the reduction in the number of characters permitted as input for registration or lookup (Section 3), and even the proposed changes in handling of right to left strings [IDNA2008-Bidi]. IDNA2008 does not use Nameprep or Stringprep at all, so there are no side-effect changes to other protocols.

It is particularly important to keep IDNA processing separate from processing for various security protocols because some of the constraints that are necessary for smooth and comprehensible use of IDNs may be unwanted or undesirable in other contexts. For example, the criteria for good passwords or passphrases are very different from those for desirable IDNs: passwords should be hard to guess, while domain names should normally be easily memorable. Similarly,

internationalized SCSI identifiers and other protocol components are likely to have different requirements than IDNs.

7.6. The Symbol Question

One of the major differences between this specification and the original version of IDNA is that the original version permitted non-letter symbols of various sorts, including punctuation and line-drawing symbols, in the protocol. They were always discouraged in practice. In particular, both the "IESG Statement" about IDNA and all versions of the ICANN Guidelines specify that only language characters be used in labels. This specification disallows symbols entirely. There are several reasons for this, which include:

1. As discussed elsewhere, the original IDNA specification assumed that as many Unicode characters as possible should be permitted, directly or via mapping to other characters, in IDNs. This specification operates on an inclusion model, extrapolating from the original "hostname" rules (LDH, see [IDNA2008-Defs]) -- which have served the Internet very well -- to a Unicode base rather than an ASCII base.

2. Symbol names are more problematic than letters because there may be no general agreement on whether a particular glyph matches a symbol; there are no uniform conventions for naming; variations such as outline, solid, and shaded forms may or may not exist; and so on. As just one example, consider a "heart" symbol as it might appear in a logo that might be read as "I love...". While the user might read such a logo as "I love..." or "I heart...", considerable knowledge of the coding distinctions made in Unicode is needed to know that there more than one "heart" character (e.g., U+2665, U+2661, and U+2765) and how to describe it. These issues are of particular importance if strings are expected to be understood or transcribed by the listener after being read out loud.

3. Design of a screen reader used by blind Internet users who must listen to renderings of IDN domain names and possibly reproduce them on the keyboard becomes considerably more complicated when

the names of characters are not obvious and intuitive to anyone familiar with the language in question.

4. As a simplified example of this, assume one wanted to use a "heart" or "star" symbol in a label. This is problematic because those names are ambiguous in the Unicode system of naming (the actual Unicode names require far more qualification). A user or would-be registrant has no way to know -- absent careful study of the code tables -- whether it is ambiguous (e.g., where there are multiple "heart" characters) or not. Conversely, the user seeing the hypothetical label doesn't know whether to read it -- try to transmit it to a colleague by voice -- as "heart", as "love", as

"black heart", or as any of the other examples below.

5. The actual situation is even worse than this. There is no possible way for a normal, casual, user to tell the difference between the hearts of U+2665 and U+2765 and the stars of U+2606 and U+2729 or the without somehow knowing to look for a distinction. We have a white heart (U+2661) and few black hearts. Consequently, describing a label as containing a heart is hopelessly ambiguous: we can only know that it contains one of several characters that look like hearts or have "heart" in their names. In cities where "Square" is a popular part of a location name, one might well want to use a square symbol in a label as well and there are far more squares of various flavors in Unicode than there are hearts or stars.

The consequence of these ambiguities is that symbols are a very poor basis for reliable communication. Consistent with this conclusion, the Unicode standard recommends that strings used in identifiers not contain symbols or punctuation [Unicode-UAX31]. Of course, these difficulties with symbols do not arise with actual pictographic languages and scripts which would be treated like any other language characters; the two should not be confused.

7.7. Migration Between Unicode Versions: Unassigned Code Points

In IDNA2003, labels containing unassigned code points are looked up on the assumption that, if they appear in labels and can be mapped and then resolved, the relevant standards must have changed and the registry has properly allocated only assigned values.

In the protocol described in these documents, strings containing unassigned code points must not be either looked up or registered. In summary, the status of an unassigned character with regard to the DISALLOWED, PROTOCOL-VALID, and CONTEXTUAL RULE REQUIRED categories cannot be evaluated until a character is actually assigned and known. There are several reasons for this, with the most important ones being:

- Tests involving the context of characters (e.g., some characters being permitted only adjacent to others of specific types) and integrity tests on complete labels are needed. Unassigned code points cannot be permitted because one cannot determine whether particular code points will require contextual rules (and what those rules should be) before characters are assigned to them and the properties of those characters fully understood.

- It cannot be known in advance, and with sufficient reliability, whether a newly-assigned code point will be associated with a

  character that would be disallowed by the rules in [IDNA2008-Tables] (such as a compatibility character). In IDNA2003, since there is no direct dependency on NFKC (many of the entries in Stringprep's tables are based on NFKC, but IDNA2003 depends only on Stringprep), allocation of a compatibility character might produce some odd situations, but it would not be a problem. In IDNA2008, where compatibility characters are DISALLOWED unless character-specific exceptions are made, permitting strings containing unassigned characters to be looked up would violate the principle that characters in DISALLOWED are not looked up.

- The Unicode Standard specifies that an unassigned code point normalizes (and, where relevant, case folds) to itself. If the code point is later assigned to a character, and particularly if the newly-assigned code point has a combining class that determines its placement relative to other combining characters, it could normalize to some other code point or sequence.

It is possible to argue that the issues above are not important and that, as a consequence, it is better to retain the principle of looking up labels even if they contain unassigned characters because

all of the important scripts and characters have been coded as of Unicode 5.1 and hence unassigned code points will be assigned only to obscure characters or archaic scripts. Unfortunately, that does not appear to be a safe assumption for at least two reasons. First, much the same claim of completeness has been made for earlier versions of Unicode. The reality is that a script that is obscure to much of the world may still be very important to those who use it. Cultural and linguistic preservation principles make it inappropriate to declare the script of no importance in IDNs. Second, we already have counterexamples in, e.g., the relationships associated with new Han characters being added (whether in the BMP or in Unicode Plane 2).

Independent of the technical transition issues identified above, it can be observed that any addition of characters to an existing script to make it easier to use or to better accommodate particular languages may lead to transition issues. Such changes may change the preferred form for writing a particular string, changes that may be reflected, e.g., in keyboard transition modules that would necessarily be different from those for earlier versions of Unicode where the newer characters may not exist. This creates an inherent transition problem because attempts to access labels may use either the old or the new conventions, requiring registry action whether the older conventions were used in labels or not. The need to consider transition mechanisms is inherent to evolution of Unicode to better accommodate writing systems and is independent of how IDNs are represented in the DNS or how transitions among versions of those

mechanisms occur. The requirement for transitions of this type is illustrated by the addition of Malayalam Chillu in Unicode 5.1.0.

7.8. Other Compatibility Issues

The 2003 IDNA model includes several odd artifacts of the context in which it was developed. Many, if not all, of these are potential avenues for exploits, especially if the registration process permits "source" names (names that have not been processed through IDNA and Nameprep) to be registered. As one example, since the character Eszett, used in German, is mapped by IDNA2003 into the sequence "ss" rather than being retained as itself or prohibited, a string containing that character but that is otherwise in ASCII is not really an IDN (in the U-label sense defined above) at all. After Nameprep maps the Eszett out, the result is an ASCII string and so

does not get an xn-- prefix, but the string that can be displayed to a user appears to be an IDN. The newer version of the protocol eliminates this artifact. A character is either permitted as itself or it is prohibited; special cases that make sense only in a particular linguistic or cultural context can be dealt with as localization matters where appropriate.

8. Name Server Considerations

8.1. Processing Non-ASCII Strings

Existing DNS servers do not know the IDNA rules for handling non-ASCII forms of IDNs, and therefore need to be shielded from them. All existing channels through which names can enter a DNS server database (for example, master files (as described in RFC 1034) and DNS update messages [RFC2136]) are IDN-unaware because they predate IDNA. Other sections of this document provide the needed shielding by ensuring that internationalized domain names entering DNS server databases through such channels have already been converted to their equivalent ASCII A-label forms.

Because of the distinction made between the algorithms for Registration and Lookup in [IDNA2008-Protocol] (a domain name containing only ASCII codepoints can not be converted to an A-label), there can not be more than one A-label form for any given U-label.

As specified in RFC 2181 [RFC2181], the DNS protocol explicitly allows domain labels to contain octets beyond the ASCII range (0000..007F), and this document does not change that. However, although the interpretation of octets 0080..00FF is well-defined in the DNS, many application protocols support only ASCII labels and there is no defined interpretation of these non-ASCII octets as

characters and, in particular, no interpretation of case-independent matching for them (see, e.g., [RFC4343]). If labels containing these octets are returned to applications, unpredictable behavior could result. The A-label form, which cannot contain those characters, is the only standard representation for internationalized labels in the DNS protocol.

8.2. DNSSEC Authentication of IDN Domain Names

DNS Security (DNSSEC) [RFC2535] is a method for supplying cryptographic verification information along with DNS messages. Public Key Cryptography is used in conjunction with digital signatures to provide a means for a requester of domain information to authenticate the source of the data. This ensures that it can be traced back to a trusted source, either directly or via a chain of trust linking the source of the information to the top of the DNS hierarchy.

IDNA specifies that all internationalized domain names served by DNS servers that cannot be represented directly in ASCII MUST use the A-label form. Conversion to A-labels MUST be performed prior to a zone being signed by the private key for that zone. Because of this ordering, it is important to recognize that DNSSEC authenticates a domain name containing A-labels or conventional LDH-labels, not U-labels. In the presence of DNSSEC, no form of a zone file or query response that contains a U-label may be signed or the signature validated.

One consequence of this for sites deploying IDNA in the presence of DNSSEC is that any special purpose proxies or forwarders used to transform user input into IDNs must be earlier in the lookup flow than DNSSEC authenticating nameservers for DNSSEC to work.

8.3. Root and other DNS Server Considerations

IDNs in A-label form will generally be somewhat longer than current domain names, so the bandwidth needed by the root servers is likely to go up by a small amount. Also, queries and responses for IDNs will probably be somewhat longer than typical queries historically, so EDNS0 [RFC2671] support may be more important (otherwise, queries and responses may be forced to go to TCP instead of UDP).

9. Internationalization Considerations

DNS labels and fully-qualified domain names provide mnemonics that assist in identifying and referring to resources on the Internet. IDNs expand the range of those mnemonics to include those based on

languages and character sets other than Western European and Roman-

derived ones. But domain "names" are not, in general, words in any language. The recommendations of the IETF policy on character sets and languages, (BCP 18 [RFC2277]) are applicable to situations in which language identification is used to provide language-specific contexts. The DNS is, by contrast, global and international and ultimately has nothing to do with languages. Adding languages (or similar context) to IDNs generally, or to DNS matching in particular, would imply context dependent matching in DNS, which would be a very significant change to the DNS protocol itself. It would also imply that users would need to identify the language associated with a particular label in order to look that label up. That knowledge is generally not available because many labels are not words in any language and some may be words in more than one.

10. IANA Considerations

This section gives an overview of IANA registries required for IDNA. The actual definitions of, and specifications for, the first two, which must be newly-created for IDNA2008, appear in [IDNA2008-Tables]. This document describes the registries but does not specify any IANA actions.

10.1. IDNA Character Registry

The distinction among the major categories "UNASSIGNED", "DISALLOWED", "PROTOCOL-VALID", and "CONTEXTUAL RULE REQUIRED" is made by special categories and rules that are integral elements of [IDNA2008-Tables]. While not normative, an IANA registry of characters and scripts and their categories, updated for each new version of Unicode and the characters it contains, will be convenient for programming and validation purposes. The details of this registry are specified in [IDNA2008-Tables].

10.2. IDNA Context Registry

IANA will create and maintain a list of approved contextual rules for characters that are defined in the IDNA Character Registry list as requiring a Contextual Rule (i.e., the types of rule described in Section 3.1.2). The details for those rules appear in [IDNA2008-Tables].

10.3. IANA Repository of IDN Practices of TLDs

This registry, historically described as the "IANA Language Character Set Registry" or "IANA Script Registry" (both somewhat misleading terms) is maintained by IANA at the request of ICANN. It is used to

provide a central documentation repository of the IDN policies used by top level domain (TLD) registries who volunteer to contribute to it and is used in conjunction with ICANN Guidelines for IDN use.

It is not an IETF-managed registry and, while the protocol changes specified here may call for some revisions to the tables, these specifications have no direct effect on that registry and no IANA action is required as a result.

11. Security Considerations

11.1. General Security Issues with IDNA

This document is purely explanatory and informational and consequently introduces no new security issues. It would, of course, be a poor idea for someone to try to implement from it; such an attempt would almost certainly lead to interoperability problems and might lead to security ones. A discussion of security issues with IDNA, including some relevant history, appears in [IDNA2008-Defs].

12. Acknowledgments

The editor and contributors would like to express their thanks to those who contributed significant early (pre-WG) review comments, sometimes accompanied by text, Paul Hoffman, Simon Josefsson, and Sam Weiler. In addition, some specific ideas were incorporated from suggestions, text, or comments about sections that were unclear supplied by Vint Cerf, Frank Ellerman, Michael Everson, Asmus Freytag, Erik van der Poel, Michel Suignard, and Ken Whistler. Thanks are also due to Vint Cerf, Lisa Dusseault, Debbie Garside, and Jefsey Morfin for conversations that led to considerable improvements in the content of this document.

A meeting was held on 30 January 2008 to attempt to reconcile differences in perspective and terminology about this set of specifications between the design team and members of the Unicode Technical Consortium. The discussions at and subsequent to that meeting were very helpful in focusing the issues and in refining the specifications. The active participants at that meeting were (in alphabetic order as usual) Harald Alvestrand, Vint Cerf, Tina Dam, Mark Davis, Lisa Dusseault, Patrik Faltstrom (by telephone), Cary Karp, John Klensin, Warren Kumari, Lisa Moore, Erik van der Poel, Michel Suignard, and Ken Whistler. We express our thanks to Google for support of that meeting and to the participants for their

contributions.

Useful comments and text on the WG versions of the draft were received from many participants in the IETF "IDNABIS" WG and a number of document changes resulted from mailing list discussions made by that group. Marcos Sanz provided specific analysis and suggestions that were exceptionally helpful in refining the text, as did Vint Cerf, Martin Duerst, Andrew Sullivan, and Ken Whistler. Lisa Dusseault provided extensive editorial suggestions during the spring of 2009, most of which were incorporated.

13. Contributors

While the listed editor held the pen, the core of this document and the initial WG version represents the joint work and conclusions of an ad hoc design team consisting of the editor and, in alphabetic order, Harald Alvestrand, Tina Dam, Patrik Faltstrom, and Cary Karp. Considerable material describing mapping principles has been incorporated from a draft of [IDNA2008-Mapping] by Pete Resnick and Paul Hoffman. In addition, there were many specific contributions and helpful comments from those listed in the Acknowledgments section and others who have contributed to the development and use of the IDNA protocols.

14. References

14.1. Normative References

[ASCII] American National Standards Institute (formerly United States of America Standards Institute), "USA Code for Information Interchange", ANSI X3.4-1968, 1968.

ANSI X3.4-1968 has been replaced by newer versions with slight modifications, but the 1968 version remains definitive for the Internet.

[IDNA2008-Bidi]
Alvestrand, H. and C. Karp, "An updated IDNA criterion for right to left scripts", August 2009, <https://

datatracker.ietf.org/drafts/draft-ietf-idnabis-bidi/>.

[IDNA2008-Defs]
Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", August 2009, <https://datatracker.ietf.org/drafts/draft-ietf-idnabis-defs/>.

[IDNA2008-Protocol]

Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", August 2009, <https://datatracker.ietf.org/drafts/draft-ietf-idnabis-protocol/>.

[IDNA2008-Tables]
Faltstrom, P., "The Unicode Code Points and IDNA", August 2009, <https://datatracker.ietf.org/drafts/draft-ietf-idnabis-tables/>.

A version of this document is available in HTML format at http://stupid.domain.name/idnabis/draft-ietf-idnabis-tables-06.html

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, March 2003.

[Unicode-UAX15]
The Unicode Consortium, "Unicode Standard Annex #15: Unicode Normalization Forms", March 2008, <http://www.unicode.org/reports/tr15/>.

[Unicode51]
The Unicode Consortium, "The Unicode Standard, Version 5.1.0", 2008.

defined by: The Unicode Standard, Version 5.0, Boston, MA, Addison-Wesley, 2007, ISBN 0-321-48091-0, as amended by

Unicode 5.1.0
(http://www.unicode.org/versions/Unicode5.1.0/).

14.2. Informative References

[BIG5] Institute for Information Industry of Taiwan, "Computer Chinese Glyph and Character Code Mapping Table, Technical Report C-26", 1984.

There are several forms and variations and a closely-related standard, CNS 11643. See the discussion in Chapter 3 of Lunde, K., CJKV Information Processing, O'Reilly & Associates, 1999

[GB18030] "Chinese National Standard GB 18030-2000: Information Technology -- Chinese ideograms coded character set for

information interchange -- Extension for the basic set.", 2000.

[IDNA2008-Mapping]
Resnick, P., "Mapping Characters in IDNA", August 2009, <https://datatracker.ietf.org/drafts/draft-ietf-idnabis-mapping/>.

[RFC0810] Feinler, E., Harrenstien, K., Su, Z., and V. White, "DoD Internet host table specification", RFC 810, March 1982.

[RFC0952] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, October 1985.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

[RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)",

RFC 2136, April 1997.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, July 1997.

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[RFC2535] Eastlake, D., "Domain Name System Security Extensions", RFC 2535, March 1999.

[RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.

[RFC2673] Crawford, M., "Binary Labels in the Domain Name System", RFC 2673, August 1999.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC3454] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")", RFC 3454,

December 2002.

[RFC3491] Hoffman, P. and M. Blanchet, "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)", RFC 3491, March 2003.

[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, April 2004.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

[RFC4290] Klensin, J., "Suggested Practices for Registration of Internationalized Domain Names (IDN)", RFC 4290, December 2005.

[RFC4343] Eastlake, D., "Domain Name System (DNS) Case Insensitivity Clarification", RFC 4343, January 2006.

[RFC4690] Klensin, J., Faltstrom, P., Karp, C., and IAB, "Review and Recommendations for Internationalized Domain Names (IDNs)", RFC 4690, September 2006.

[RFC4713] Lee, X., Mao, W., Chen, E., Hsu, N., and J. Klensin, "Registration and Administration Recommendations for Chinese Domain Names", RFC 4713, October 2006.

[Unicode-Security] The Unicode Consortium, "Unicode Technical Standard #39: Unicode Security Mechanisms", August 2008, <http://www.unicode.org/reports/tr39/>.

[Unicode-UAX31] The Unicode Consortium, "Unicode Standard Annex #31: Unicode Identifier and Pattern Syntax", March 2008, <http://www.unicode.org/reports/tr31/>.

[Unicode-UTR36] The Unicode Consortium, "Unicode Technical Report #36: Unicode Security Considerations", July 2008, <http://www.unicode.org/reports/tr36/>.

Appendix A. Change Log

[[RFC Editor: Please remove this appendix.]]

A.1. Changes between Version -00 and Version -01 of draft-ietf-idnabis-rationale

- Clarified the U-label definition to note that U-labels must contain at least one non-ASCII character. Also clarified the relationship among label types.

- Rewrote the discussion of Labels in Registration (Section 7.1.2) and related text about IDNA-validity (in the "Defs" document as of -04 of this one) to narrow its focus and remove more general restrictions. Added a temporary note in line to explain the

situation.

- Changed the "IDNA uses Unicode" statement to focus on compatibility with IDNA2003 and avoid more general or controversial assertions.

- Added a discussion of examples to Section 7.1

- Made a number of other small editorial changes and corrections suggested by Mark Davis.

- Added several more discussion anchors and notes and expanded or updated some existing ones.

A.2. Version -02

- Trimmed change log, removing information about pre-WG drafts.

- Adjusted discussion of Contextual Rules to match the new location of the tables and some conceptual material.

- Rewrote the material on preprocessing somewhat.

- Moved the material about relationships with IDNA2003 to be part of a single section on transitions.

- Removed several placeholders and made editorial changes in accordance with decisions made at IETF 72 in Dublin and not disputed on the mailing list.

A.3. Version -03

This special update to the Rationale document is intended to try to get the discussion of what is normative or not under control. While the IETF does not normally annotate individual sections of documents with whether they are normative or not, concerns that we don't know which is which, claims that some material is normative that would be

problematic if so classified, etc., argue that we should at least be able to have a clear discussion on the subject.

Two annotations have been applied to sections that might reasonably

be considered normative. One annotation is based on the list of sections in Mark Davis's note of 29 September (http://www.alvestrand.no/pipermail/idna-update/2008-September/002667.html). The other is based on an elaboration of John Klensin's response on 7 October (http://www.alvestrand.no/pipermail/idna-update/2008-October/002691.html). These should just be considered two suggestions to illuminate and, one hopes, advance the Working Group's discussions.

Some additional editorial changes have been made, but they are basically trivial. In the editor's judgment, it is not possible to make significantly more progress with this document until the matter of document organization is settled.

A.4. Version -04

- Definitional and other normative material moved to new document (draft-ietf-idnabis-defs). Version -03 annotations removed.

- Material on differences between IDNA2003 and IDNA2008 moved to an appendix in Protocol.

- Material left over from the origins of this document as a preliminary proposal has been removed or rewritten.

- Changes made to reflect consensus call results, including removing several placeholder notes for discussion.

- Added more material, including discussion of historic scripts, to Section 3.2 on registration policies.

- Added a new section (Section 7.2) to contain specific discussion of handling of characters that are interpreted differently in input to IDNA2003 and 2008.

- Some material, including this section/appendix, rearranged.

A.5. Version -05

- Many small editorial changes, including changes to eliminate the last vestiges of what appeared to be 2119 language (upper-case MUST, SHOULD, or MAY) and small adjustments to terminology.

A.6. Version -06

- Removed Security Considerations material and pointed to Defs, where it now appears as of version 05.

- Started changing uses of "IDNA2008" in running text to "in these specifications" or the equivalent. These documents are titled simply "IDNA"; once they are standardized, "the current version" may be a more appropriate reference than one containing a year. As discussed on the mailing list, we can and should discuss how to refer to these documents at an appropriate time (e.g., when we know when we will be finished) but, in the interim, it seems appropriate to simply start getting rid of the version-specific terminology where it can naturally be removed.

- Additional discussion of mappings, etc., especially for case-sensitivity.

- Clarified relationship to base DNS specifications.

- Consolidated discussion of lookup of unassigned characters.

- More editorial fine-tuning.

A.7. Version -07

- Revised terminology by adding terms: NR-LDH-label, Invalid-A-label (or False-A-label), R-LDH-label, valid IDNA-label in Section 1.3.2.

- Moved the "name server considerations" material to this document from Protocol because it is non-normative and not part of the protocol itself.

- To improve clarity, redid discussion of the reasons why looking up unassigned code points is prohibited.

- Editorial and other non-substantive corrections to reflect earlier errors as well as new definitions and terminology.

A.8. Version -08

- Slight revision to "contextual" discussion (Section 3.1.2) and moving it to a separate subsection, rather than under "PVALID", for better parallelism with Tables. Also reflected Mark's comments about the limitations of the approach.

- o Added placeholder notes as reminders of where references to the other documents need Section numbers. More of these will be added as needed (feel free to identify relevant places), but the actual section numbers will not be inserted until the documents are completely stable, i.e., on their way to the RFC Editor.

A.9. Version -09

- o Small editorial changes to clarify transition possibilities.

- o Small clarification to the description of DNS "exact match".

- o Added discussion of adding characters to an existing script to the discussion of unassigned code point transitions in Section 7.7.

- o Tightened up the discussion of non-ASCII string processing (Section 8.1) slightly.

- o Removed some placeholders and comments that have been around long enough to be considered acceptable or that no longer seem necessary for other reasons.

A.10. Version -10

- o Extensive editorial improvements, mostly due to suggestions from Lisa Dusseault.

- o Changes required for the new "mapping" approach and document have, in general, not been incorporated despite several suggestions. The editor intends to wait until the mapping model is stable, or at least until -11 of this document, before trying to incorporate those suggestions.

A.11. Version -11

- o Several placeholders for additional material or editing have been removed since no comments have been received.

- o Updated references.

o Corrected an apparent patching error in Section 1.6 and another one in Section 4.3.

o Adjusted several sections that had not properly reflected removal of the material that is now in the Definitions document and removed an unnecessary one.

o New material added to Section 3.2 about registration policy issues to reflect discussions on the mailing list.

o Incorporated mapping material from the former "Architectural Principles" of version -01 of the Mapping draft into Section 6 and removed most of the prior mapping material and explanations.

o Eliminated the former Section 7.3 ("More Flexibility in User Agents"), moving its material into Section 4.2. The replacement section is basically a placeholder to retain the mapping issues as one of the migration topics. Note that this item and the previous one involve considerable text, so people should check things carefully.

o Corrected several typographical and editorial errors that don't fall into any of the above categories.

A.12. Version -12

o Got rid of the term "IDNA-valid". It no longer appears in Definitions and we didn't really need the extra term. Where the concept was needed, the text now says "valid under IDNA" or equivalent.

o Adjusted Acknowledgments to remove Mark Davis's name, per his request and advice from IETF Trust Counsel.

o Incorporated other changes from WG Last Call.

o Small typographical and editorial corrections.

Author's Address

John C Klensin
1770 Massachusetts Ave, Ste 322
Cambridge, MA 02140
USA

Phone: +1 617 245 1457
Email: john+ietf@jck.com