Network Working Group
Internet-Draft
Expires: May 21, 2008

J. Klensin, Ed.

November 18, 2007

Internationalizing Domain Names for Applications (IDNA): Issues and Rationale

draft-klensin-idnabis-issues-05.txt

Status of this Memo

Copyright Notice

Abstract

A recent IAB report identified issues that have been raised with Internationalized Domain Names (IDNs). Some of these issues require tuning of the existing protocols and the tables on which they depend. Based on intensive discussion by an informal design team, this document provides an overview some of the proposals that are being made, provides explanatory material for them and then further explains some of the issues that have been encountered.

Table of Contents

1. Introduction

1.1. Context and Overview

A recent IAB report [RFC4690] identified issues that have been raised with Internationalized Domain Names (IDNs) and the associated standards. Those standards are known as Internationalized Domain Names in Applications (IDNA), taken from the name of the highest level standard within that group (see Section 1.5). Based on discussion of those issues and their impact, some of these standards now require tuning the existing protocols and the tables on which they depend. This document further explains, based on the results of some intensive discussions by an informal design team, on a mailing list, and in broader discussions, some of the issues that have been encountered. It also provides an overview of the proposals that are being made and explanatory material for them. Additional explanatory material for other proposals will appear with the associated documents.

This document begins with a discussion of the original and new IDNA models and the general differences in strategy between the original version of IDNA and the proposed new version. It continues with a description of specific changes that are needed and issues that the design must address, including some that were not explicitly addressed in RFC 4690.

1.2. Discussion Forum

[[anchor4: RFC Editor: please remove this section.]]

This work is being discussed on the mailing list idna-update@alvestrand.no

1.3. Objectives

The intent of the IDNA revision effort, and hence of this document and the associated ones, is to increase the usability and effectiveness of internationalized domain names (IDNs) while preserving or strengthening the integrity of references that use them. The original "hostname" (LDH) character definitions (see, e.g., [RFC0810]) struck a balance between the creation of useful mnemonics and the introduction of parsing problems or general confusion in the contexts in which domain names are used. Our objective is to preserve that balance while expanding the character repertoire to include extended versions of Roman-derived scripts and scripts that are not Roman in origin. No work of this sort will be able to completely eliminate sources of visual or textual confusion: such confusion exists even under the original rules. However, one

can hope, through the application of different techniques at different points (see Section 5.2), to keep problems to an acceptable minimum. One consequence of this general objective is that the desire of some user or marketing community to use a particular string --whether the reason is to try to write sentences of particular languages in the DNS, to express a facsimile of the symbol for a brand, or for some other purpose-- is not a primary goal or even a particularly important one.

1.4. Applicability and Function of IDNA

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 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 the user interface.

A great deal of the discussion of IDN solutions has focused on transition issues and how IDN 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 depend on user applications, resolvers, and DNS servers being updated in order for a user to use 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, no changes are needed to the DNS protocol or any DNS servers or the resolvers on user's computers.

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 the service offered by DNS to the applications. Instead, the applications (and, by implication, the users) continue to see an exact-match lookup service. Either there is a single exactly-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.

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 rudimentary use of IDNs in applications by using the ASCII representation of the non-ASCII name labels. While such names are user-unfriendly to read and type, and hence not optimal for user input, they allow (for instance) replying to email and clicking on URLs even though the domain name displayed is incomprehensible to the user. In order to allow user-friendly input and output of the IDNs, the applications need to be modified to conform to this specification.

IDNA uses the Unicode character repertoire, which avoids the significant delays that would be inherent in waiting for a different and specific character set be defined for IDN purposes, presumably by some other standards developing organization.

1.5. Terminology

1.5.1. Documents and Standards

This document uses the term "IDNA2003" to refer to the set of standards that make up and support the version of IDNA published in 2003, i.e., those commonly known as the IDNA base specification [RFC3490], Nameprep [RFC3491], Punycode [RFC3492], and Stringprep [RFC3454]. In this document, those names are used to refer, conceptually, to the individual documents, with the base IDNA specification called just "IDNA".

The term "IDNA200x" is used to refer to a possible new version of IDNA without specifying which particular documents would be affected. While more common IETF usage might refer to the successor document(s) as "IDNAbis", this document uses that term, and similar ones, to refer to successors to the individual documents, e.g., "IDNAbis" is a synonym for the specific successor to RFC3490, or "RFC3490bis". See also Section 4.

1.5.2. Terminology about Characters and Character Sets

A code point is an integer value associated with a character in a coded character set.

Unicode [Unicode50] is a coded character set containing tens of thousands of characters. A single Unicode code point is denoted by "U+" followed by four to six hexadecimal digits, while a range of Unicode code points is denoted by two hexadecimal numbers separated by "..", with no prefixes.

ASCII means US-ASCII [ASCII], a coded character set containing 128 characters associated with code points in the range 00..7F. Unicode

may be thought of as an extension of ASCII: it includes all the ASCII characters and associates them with equivalent code points.

1.5.3. DNS-related Terminology

When discussing the DNS, this document generally assumes the terminology used in the DNS specifications [RFC1034] [RFC1035]. The terms "lookup" and "resolution" are used interchangeably and the process or application component that performs DNS resolution is called a "resolver". The process of placing an entry into the DNS is referred to as "registration" paralleling common contemporary usage in other contexts.

The term "LDH code points" is defined in this document to mean the code points associated with ASCII letters, digits, and the hyphen-minus; that is, U+002D, 30..39, 41..5A, and 61..7A. "LDH" is an abbreviation for "letters, digits, hyphen".

The base DNS specifications [RFC1034] [RFC1035] discuss "domain names" and "host names", but many people and sections of these specifications use the terms interchangeably. Further, because those documents were not terribly clear, many people who are sure they know the exact definitions of each of these terms disagree on the definitions. In this document the term "domain name" is used in general. This document explicitly cites those documents whenever referring to the host name syntax restrictions defined therein. The remaining definitions in this subsection are essentially a review.

A label is an individual part of a domain name. Labels are usually shown separated by dots; for example, the domain name "www.example.com" is composed of three labels: "www", "example", and "com". (The zero-length root label described in [RFC1123], which can be explicit as in "www.example.com." or implicit as in "www.example.com", is not considered a label in this specification.) IDNA extends the set of usable characters in labels that are text. For the rest of this document, the term "label" is shorthand for "text label", and "every label" means "every text label".

1.5.4. Terminology Specific to IDNA

Some of the terminology used in describing IDNs in the IDNA2003 context has been a source of confusion. This section defines some new terminology to reduce dependence on the problematic terms and definitions that appears in RFC 3490.

1.5.4.1. Terms for IDN Label Codings

1.5.4.1.1. IDNA-valid strings, A-label, and U-label

To improve clarity, this document introduces three new terms. A string is "IDNA-valid" if it meets all of the requirements of this specification for an IDNA label. It may be either an "A-label" or a "U-label", and it is expected that specific reference will be made to the form appropriate to any context in which the distinction is important. An "A-label" is the ASCII-Compatible Encoding (ACE) form of an IDNA-valid string. It must be a complete label and valid as the output of ToASCII, regardless of how it is actually produced. This means, by definition, that every A-label will begin with the IDNA ACE prefix, "xn--", followed by a string that is a valid output of the Punycode algorithm and hence a maximum of 59 ASCII characters in length. The prefix and string together must conform to all requirements for a label that can be stored in the DNS including conformance to the LDH rule. A "U-label" is an IDNA-valid string of Unicode-coded characters that is a valid output of performing ToUnicode on an A-label, again regardless of how the label is actually produced. A Unicode string that cannot be generated by decoding a valid A-label is not a valid U-label. [IDNA200X-protocol] specifies the conversions between U-labels and A-labels.

Any rules or conventions that apply to DNS labels in general, such as rules about lengths of strings, apply to whichever of the U-label or A-label would be more restrictive. The exception to this, of course, is that the restriction to ASCII characters does not apply to the U-label.

A different way to look at these terms, which may be more clear to some readers, is that U-labels, A-labels, and LDH-labels are disjoint categories that, together, make up the forms of legitimate strings for use in domain names that describe hosts. Of the three, only A-labels and LDH-labels can actually appear in DNS zone files or queries; U-labels can appear, along with those two, in presentation and user interface forms and in selected protocols other than the DNS ones themselves. Strings that do not conform to the rules for one of these three categories and, in particular, strings that contain "-" in the third or fourth character position but are

- not A-labels or
- that cannot be processed as U-labels or A-labels as described in these specifications,

are invalid as labels in domain names that identify Internet hosts or similar resources.

1.5.4.1.2. LDH-label and Internationalized Label

In the hope of further clarifying discussions about IDNs, this document uses the term "LDH-label" strictly to refer to an all-ASCII label that obeys the "hostname" (LDH) conventions and that is not an IDN. In other words, the categories "U-label", "A-label", and "LDH-label" are disjoint, with only the first two referring to IDNs. When such a term is needed, an "internationalized label" is one that is a member of the union of those three categories. There are some standardized DNS label formats, such as those for service location (SRV) records [RFC2782] that do not fall into any of the three categories and hence are not internationalized labels.

1.5.4.2. Equivalence

In IDNA, equivalence of labels is defined in terms of the A-labels. If the A-labels are equal in a case-independent comparison, then the labels are considered equivalent, no matter how they are represented. Traditional LDH labels already have a notion of equivalence: within that list of characters, upper case and lower case are considered equivalent. The IDNA notion of equivalence is an extension of that older notion. Equivalent labels in IDNA are treated as alternate forms of the same label, just as "foo" and "Foo" are treated as alternate forms of the same label.

1.5.4.3. ACE prefix

The "ACE prefix" is defined in this document to be a string of ASCII characters "xn--" that appears at the beginning of every A-label. "ACE" stands for "ASCII-Compatible Encoding".

1.5.4.4. Domain name slot

A "domain name slot" is defined in this document to be a protocol element or a function argument or a return value (and so on) explicitly designated for carrying a domain name. Examples of domain name slots include: the QNAME field of a DNS query; the name argument of the gethostbyname() library function; the part of an email address following the at-sign (@) in the From: field of an email message header; and the host portion of the URI in the src attribute of an HTML <IMG> tag. General text that just happens to contain a domain name is not a domain name slot. For example, a domain name appearing in the plain text body of an email message is not occupying a domain name slot.

An "IDN-aware domain name slot" is defined in this document to be a domain name slot explicitly designated for carrying an internationalized domain name as defined in this document. The

designation may be static (for example, in the specification of the protocol or interface) or dynamic (for example, as a result of negotiation in an interactive session).

An "IDN-unaware domain name slot" is defined in this document to be any domain name slot that is not an IDN-aware domain name slot. Obviously, this includes any domain name slot whose specification predates IDNA.

1.5.5. Punycode is an Algorithm, not a Name

There has been some confusion about whether a "Punycode string" does or does not include the prefix and about whether it is required that such strings could have been the output of ToASCII (see RFC 3490, Section 4 [RFC3490]). This specification discourages the use of the term "Punycode" to describe anything but the encoding method and algorithm of [RFC3492]. The terms defined above are preferred as much more clear than terms such as "Punycode string".

1.5.6. Other Terminology Issues

The document departs from historical DNS terminology and usage in one important respect. Over the years, the community has talked very casually about "names" in the DNS, beginning with calling it "the domain name system". That terminology is fine in the very precise sense that the identifiers of the DNS do provide names for objects and addresses. But, in the context of IDNs, the term has introduced some confusion, confusion that has increased further as people have begun to speak of DNS labels in terms of the words or phrases of various natural languages.

Historically, many, perhaps most, of the "names" in the DNS have just been mnemonics to identify some particular concept, object, or organization. They are typically derived from, or 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.

An "internationalized domain name" (IDN) is a domain name that may contain one or more A-labels or U-labels, as appropriate, instead of LDH labels. This implies that every conventional domain name is an

IDN (which implies that it is possible for a name to be an IDN without it containing any non-ASCII characters). This document does not attempt to define an "internationalized host name". Just as has been the case with ASCII names, some DNS zone administrators may impose restrictions, beyond those imposed by DNS or IDNA, on the characters or strings that may be registered as labels in their zones. Such restrictions have no effect on the syntax or semantics of DNS protocol messages; a query for a name that matches no records will yield the same response regardless of the reason why it is not in the zone. Clients issuing queries or interpreting responses cannot be assumed to have any knowledge of zone-specific restrictions or conventions.

2. The Original (2003) IDNA Model

IDNA is a client-side protocol, i.e., almost all of the processing is performed by the client. The strings that appear in, and are resolved by, the DNS conform to the traditional rules for the naming of hosts, and consist of ASCII letters, digits, and hyphens. This approach permits IDNA to be deployed without modifications to the DNS itself. That, in turn, avoids both having to upgrade the entire Internet to support IDNs and needing to incur the unknown risks to deployed systems of DNS structural or design changes especially if those changes need to be deployed all at the same time.

This section contains a summary of the model underlying IDNA2003. It is approximate and is not a substitute for reading and understanding the actual specification document [RFC3490] and the documents on which it depends. The summary is not intended to be completely balanced. It emphasizes some characteristics of IDNA2003 that are particularly important to understanding the nature of the proposed changes.

The original IDNA specifications have the logical flow in domain name registration and resolution outlined in the balance of this section. They are not defined this way; instead, the steps are presented here for convenience in comparison to what is being proposed in this document and the associated ones. In particular, IDNA2003 does not make as strong a distinction between procedures for registration and those for resolution as the ones suggested in Section 3 and Section 5.1.

The IDNA2003 specification explicitly includes the equivalents of the steps in Section 2.2, Section 2.3, and Section 2.5 below. While the other steps are present --either inside the protocol or presumed to be performed before or after it-- they are not discussed explicitly. That omission has been a source of confusion. Another source has

been definition of IDNA2003 as an algorithm, expressed partially in prose and partially in pseudo code and tables. The steps below follow the more traditional IETF practice: the functions are specified, rather than the algorithms. The breakdown into steps is for clarity of explanation; any implementation that produces the same result with the same inputs is conforming.

2.1. Proposed label

The registrant submits a request for an IDN or the user attempts to look up an IDN. The registrant or user typically produces the request string by keyboard entry of a character sequence. That sequence is validated only on the basis of its displayed appearance, without knowledge of the character coding used for its internal representation or other local details of the way the operating system processes it. This string is converted to Unicode if necessary. IDNA2003 assumes that the conversion is straightforward enough not to be considered by the protocol.

2.2. Permitted Character Identification

The Unicode string is examined to prohibit characters that IDNA does not permit in input. The list of excluded characters is quite limited because IDNA2003 permits almost all Unicode characters to be used as input, with many of them mapped into others.

2.3. Character Mappings

The label string is processed through the Nameprep [RFC3491] profile of the Stringprep [RFC3454] tables and procedure. Among other things, these procedures apply the Unicode normalization procedure NFKC [Unicode-UAX15] which converts compatibility characters to their base forms and resolves the different ways in which some characters can be represented in Unicode into a canonical form. In IDNA2003, one-way case mapping was also performed, partially simulating the query-time folding operation that the DNS provides for ASCII strings.

2.4. Registry Restrictions

Registries at all levels of the DNS, not just the top level, are expected to establish policies about the labels that may be registered and for the processes associated with that action (see the discussion of guidelines and statements in [RFC4690]). Such restrictions have always existed in the DNS and have always been applied at registration time, with the most notable example being enforcement of the hostname (LDH) convention itself. For IDNs, the restrictions to be applied are not an IETF matter except insofar as they derive from restrictions imposed by application protocols (e.g.,

email has always required a more restricted syntax for domain names than the restrictions of the DNS itself). Because these are restrictions on what can be registered, it is not generally necessary that they be global. If a name is not found on resolution, it is not relevant whether it could have been registered; only that it was not registered. Registry restrictions might include prohibition of mixed-script labels or restrictions on labels permitted in a zone if certain other labels are already present. The "variant" systems discussed in [RFC3743] and [RFC4290] are examples of fairly sophisticated registry restriction models. The various sets of ICANN IDN Guidelines [ICANN-Guidelines] also suggest restrictions that might sensibly be imposed.

The string produced by the above steps is checked and processed as appropriate to local registry restrictions. Application of those registry restrictions may result in the rejection of some labels or the application of special restrictions to others.

2.5. Punycode Conversion

The resulting label (in Unicode code point character form) is processed with the Punycode algorithm [RFC3492] and converted to a form suitable for storage in the DNS (the "xn--..." form).

2.6. Lookup or Insertion in the Zone

For registration, the Punycode-encoded label is then placed in the DNS by insertion into a zone. For lookup, that label is processed according to normal DNS query procedures [RFC1035].

3. A Revised IDNA Model

One of the major goals of this work 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 themselves important motivations and design goals for this effort. The effort includes some new terminology and a revised and extended model, both covered in this section, and some more specific protocol, processing, and table modifications. Details of the latter appear in other documents (see Section 4).

3.1. Localization: The Role of the Local System and User Interface

Several issues are inherent in the application of IDNs and, indeed, almost any other system that tries to handle international characters and concepts. They range from the apparently trivial --e.g., one cannot display a character for which one does not have a font

available locally-- to the more complex and subtle. Many people have observed that internationalization is just a tool to permit effective localization while permitting some global uniformity. Issues of display, of exactly how various strings and characters are entered, and so on are inherently issues about localization and user interface design.

A protocol such as IDNA can only assume that such operations as data entry are possible. It may make some recommendations about how display might work when characters and fonts are not available, but they can only be general recommendations.

Operations for converting between local character sets and Unicode are part of this general set of user interface issues. The conversion is obviously not required at all in a Unicode-native system where no conversion is required. It may, however, involve some complexity in one that is not, especially if the elements of the local character set do not map exactly and unambiguously into Unicode characters and do so in a way that is completely stable over time. Perhaps more important, if a label being converted to a local character set contains Unicode characters that have no correspondence in that character set, the application may have to apply special, locally-appropriate, methods to avoid or reduce loss of information.

Depending on the system involved, the major difficulty may not lie in the mapping but in accurately identifying the incoming character set and then applying the correct conversion routine. It may be especially difficult when the character coding system in local use is based on conceptually different assumptions than those used by Unicode about, e.g., how different presentation or combining forms are handled. 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.

3.2. IDN Processing in the IDNA200x Model

[[anchor20: Placeholder ??? Do we need a summary of the two parts here???]]

3.2.1. Summary of Effects

Separating Domain Name Registration and Resolution in the protocol specification has one substantive impact. With IDNA2003, the tests and steps made in these two parts of the protocol are essentially identical. Separating them reflects current practice in which per-registry restrictions and special processing are applied at registration time but not on resolution. Even more important in the

longer term, it allows incremental addition of permitted character groups to avoid freezing on one particular version of Unicode.

4. IDNA200x Document List

[[anchor22: This section will need to be extensively revised or removed before publication.]]

The following documents are being produced as part of the IDNA200x effort.

- A revised version of this document, containing an overview, rationale, and conformance conditions.
- A separate document, drawn from material in early versions of this one, that explicitly updates and replaces RFC 3490 but which has most rationale material from that document moved to this one [IDNA200X-protocol].
- A document describing the "Bidi problem" with Stringprep and proposing a solution [IDNA200X-Bidi].
- A list of code points allowed in a U-label, based on Unicode 5.0 code assignments. See Section 5.
- One or more documents containing guidance and suggestions for registries (in this context, those responsible for establishing policies for any zone file in the DNS, not only those at the top or second level). The documents in this category may not all be IETF products and may be prepared and completed asynchronously with those described above.

5. Permitted Characters: An Inclusion List

This section describes the model used to establish the algorithm and character lists of [IDNA200X-Tables] and describes the names and applicability of the categories used there. Note that the inclusion of a character in one of the first three categories does not imply that it can be used indiscriminately; some characters are associated with contextual rules that must be applied as well.

5.1. A Tiered Model of Permitted Characters and Labels

Moving to an inclusion model requires a new list of characters that are permitted in IDNs. In IDNA2003, the role and utility of

characters are independent of context and fixed forever. Making those rules globally has proven impractical, partially because handling of particular characters across the languages that use a script, or the use of similar or identical-looking characters in different scripts, are less well understood than many people believed several years ago. Conversely, IDNA2003 prohibited some characters entirely to avoid dealing with some of the issues discussed here -- restrictions that were much too severe for mnemonics based on some languages.

Independently of the characters chosen (see next subsection), the theory is to divide the characters that appear in Unicode into four categories:

5.1.1. ALWAYS

Characters identified as "ALWAYS" are permitted for all uses in IDNs, but may be associated with contextual restrictions (for example, any character in this group that has a "right to left" property must be used in context with the "Bidi" rules). The presence of a character in this category implies that it has been examined and determined to be appropriate for IDN use, and that it is well-understood that contextual protocol restrictions in addition to those already specified, such as rules about the use of given characters, are not required. That, in turn, indicates that the script community relevant to that character, reflecting appropriate authorities for all of the known languages that use that script, has agreed that the script and its components are sufficiently well understood. This subsection discusses characters, rather than scripts, because it is explicitly understood that a script community may decide to include some characters of the script and not others.

Because of this condition, which requires evaluation by individual script communities of the characters suitable for use in IDNs (not just, e.g., the general stability of the scripts in which those characters are embedded) it is not feasible to define the boundary point between this category and the next one by general properties of the characters, such as the Unicode property lists.

Despite its name, the presence of a character on this list 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 [IDNA200X-protocol] and Section 5.2).

Characters that are placed in the "ALWAYS" category are never removed from it unless the code points themselves are removed from Unicode (a

condition that may never occur).

5.1.2. MAYBE

Characters that are used to write the languages of the world and that are thought of broadly as "letters" rather than, e.g., symbols or punctuation, and that have not been placed in the "ALWAYS" or "NEVER" categories (see Section 5.1.4 for the latter) belong to the "MAYBE" category. As implied above, the collection of scripts and characters in "MAYBE" has not yet been reviewed and finally approved by the script community. It is possible that they may be appropriate for general use only when special contextual rules (tests on the entire label or on adjacent characters) are identified and specified.

In general and for maximum safety, registries SHOULD confine themselves to characters from the "ALWAYS" category. However, if a registry is permitting registrations only in a small number of scripts the usage of which it is familiar with to develop rules that are safe in its own environment -- it may be entirely appropriate for it permit registrations that use characters from the "MAYBE" categories as well as the "ALWAYS" one.

Applications are expected to not treat "ALWAYS" and "MAYBE" differently with regard to name resolution ("lookup"). They may choose to provide warnings to users when labels or fully-qualified names containing characters in the "MAYBE" categories are to be presented to users.

There are actually two subcategories of MAYBE. The assignment of a character to one or the other represents an estimate of whether the character will eventually be treated as "ALWAYS" or "NEVER" (some characters may, however, remain in the "MAYBE" categories indefinitely). Since the differences between the "MAYBE" subcategories do not affect the protocol, characters may be moved back and forth between them as information and knowledge accumulates.

5.1.2.1. Subcategory MAYBE YES

These are letter, digit, or letter-like characters that are generally presumed to be appropriate in DNS labels, for which no specific in-depth script or character evaluation has been performed. The risk with characters in the "MAYBE YES" category is that it may later be discovered that contextual rules are required for their safe use with labels that otherwise contain characters from arbitrary scripts or that the characters themselves may be problematic.

5.1.2.2. Subcategory MAYBE NO

These are characters that are not letter-like, but are not excluded by some other rule. Given the general ban on characters other than letters and digits, it is likely that they will be moved to "NEVER" when their contexts are fully understood by the relevant community. However, since characters once moved to "NEVER" cannot be moved back out, conservatism about making that classification is in order.

5.1.3. CONTEXTUAL RULE REQUIRED

These characters are unsafe for general use in IDNs, typically because they are invisible in most scripts but affect format or presentation in a few others or because they are combining characters that are safe for use only in conjunction with particular characters or scripts. In order to permit them to be used at all, these characters are assigned to the category "CONTEXTUAL RULE REQUIRED" and, when adequately understood, associated with a rule. Examples of typical rules include "Must follow a character from Script XYZ", "MAY occur only if the entire label is in Script ABC", "MAY occur only if the previous and subsequent characters have the DEF 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 character in the CONTEXTUAL RULE REQUIRED category may have a null (missing) rule set in a given version of the tables. Such characters MUST NOT appear in putative labels for either registration or lookup. Of course, a later version of the tables might contain a non-null rule.

If there is a rule, it MUST be evaluated and tested on registration and SHOULD be evaluated and tested on lookup. If the test fails, the label should not be processed for registration or lookup in the DNS.

5.1.4. NEVER

Some characters are sufficiently problematic for use in IDNs that they should be excluded for both registration and lookup (i.e., conforming applications performing name resolution 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.

Of course, this category includes code points that have been removed entirely from Unicode should such characters ever occur.

Characters that are placed in the "NEVER" category are never removed from it or reclassified. If a character is classified as "NEVER" in error and the error is sufficiently problematic, the only recourse is

to introduce a new code point into Unicode and classify it as "MAYBE" or "ALWAYS" as appropriate.

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

The essence of the character rules in IDNAbis is that there is no magic bullet for any of the issues associated with a multiscript DNS. Instead, we need to have a variety of approaches that, together, constitute multiple lines of defense. The actual 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 a protocol context. Registrars 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.

5.3. A New Character List -- History

[[anchor29: RFC Editor: please delete this subsection.]]

A preliminary version of a character list that reflects the above categories has been was developed by the contributors to this document [IDNA200X-Tables]. An earlier, initial, version was developed by going through Unicode 5.0 one block and one character class at a time and determining which characters, classes, and blocks were clearly acceptable for IDNs, which one were clearly unacceptable (e.g., all blocks consisting entirely of compatibility characters and non-language symbols were excluded as were a number of character classes), and which blocks and classes were in need of further study or input from the relevant language communities. That effort was successful, but not at the level of producing a directly-useful character table. Additional iterations on the mailing list and with UTC participation largely dropped the use of Unicode blocks and focused on character classes, scripts, and properties together with understandings gained from other Unicode Consortium efforts. Those iterations have been more successful. The iterative process has led to the conclusion that the best strategy is likely to be a mixed one consisting of (i) classification into "ALWAYS" and "MAYBE YES" versus "MAYBE NO" and "NEVER" based on Unicode properties and a few exceptions and (ii) discrimination between "ALWAYS" and "MAYBE YES" and between "MAYBE NO" and "NEVER" based on script community criteria about IDN appropriateness will be needed. An alternative would involve an entirely new property specifically associated with appropriateness for IDN use, but it is not clear that is either necessary or desirable.

5.4. Understanding New Issues and Constraints

The discussion in [IDNA200X-Bidi] illustrates some areas in which more work and input is needed. Other issues are raised by the Unicode "presentation form" model and, in particular, by the need for zero-width characters in some limited cases to correctly designate those forms and by some other issues with combining characters in different contexts. It is expected that, once expert and materially-concerned parties are identified to supply contextual rules, such problems will be resolved quickly and the questioned collections of characters either added to the list of permitted characters or permanently excluded.

5.5. ALWAYS, MAYBE, and Contextual Rules

As discussed above, characters will be associated with the "ALWAYS" or "MAYBE YES" properties if they can plausibly be used in an IDN. They are classified as "MAYBE NO" if it appears unlikely that they should be used in IDNs but there is uncertainty on that point. Non-language characters and other character codes that can be identified as globally inappropriate for IDNs, such as conventional spaces and punctuation, will be assigned to "NEVER" (i.e., will never be permitted in IDNs). A character associated with "CONTEXTUAL RULE REQUIRED" is acceptable in a label if it is associated with the identifier of a contextual rule set and the test implied by the rule set is successful. If no such identifier is present in the version of the tables in use, the character is treated as roughly equivalent to "NEVER", i.e., it MUST NOT be used in either registration or lookup with that version of the tables. Because a rule set identifier may be installed in a later table version, this status is obviously not permanent. This general approach could, obviously, be implemented in several ways, not just by the exact arrangements suggested above.

The property and rule sets are used as follows:

- Systems supporting domain name resolution SHOULD attempt to resolve any label consisting entirely of characters that are in the "ALWAYS" or "MAYBE" categories, including those that have not been permanently excluded but that have not been classified with regard to whether additional restrictions are needed, i.e., they are categorized as "MAYBE YES" or "MAYBE NO". They MUST NOT attempt to resolve label strings that contain unassigned character positions or those that contain "NEVER" characters.

- Systems providing domain name registration functions MUST NOT register any label that contains characters classified as "NEVER" OR code point positions that are unassigned in the version of

Unicode they are using. If a character in a label has associated contextual rules, they MUST NOT register the label unless the conditions required by those rules are satisfied. They SHOULD NOT register labels that contain a character assigned to a "MAYBE" category.

A procedure for assigning rules to characters with the "MAYBE YES" or "MAYBE NO" property, and for assigning (or not) the property to characters assigned in future version of Unicode, is outlined under Section 11. A key part of that procedure will be specifications that make it possible to add new characters and blocks without long delays in implementation. The procedure will result in an update to existing IANA-maintained registries.

6. Issues that Any Solution Must Address

6.1. Display and Network Order

The correct treatment of domain names requires a clear distinction between Network Order (the order in which the code points are sent in protocols) and Display Order (the order in which the code points are displayed on a screen or paper). The order of labels in a domain name is discussed in [IDNA200X-Bidi]. There are, however, also questions about the order in which labels are displayed if left-to-right and right-to-left labels are adjacent to each other, especially if there are also multiple consecutive appearances of one of the types. The decision about the display order is ultimately under the control of user agents --including web browsers, mail clients, and the like-- which may be highly localized. Even when formats are specified by protocols, the full composition of an Internationalized Resource Identifier (IRI) [RFC3987] 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. User agents are not required to use those protocol-based forms directly but often do so. While display, parsing, and processing within a label is specified by the IDNA protocol and the associated documents, the relationship between fully-qualified domain names and internationalized labels is unchanged from the base DNS specifications. Comments here about such full domain names are explanatory or examples of what might be done and must not be considered normative.

Questions remain about protocol constraints implying that the overall direction of these strings will always be left-to-right (or right-to-left) for an IRI or email address, or if they even should conform to such rules. These questions also have several possible answers.

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? An IRI for clear text web access would, in network order, begin with "http://" and the characters will appear as "http://abc.def" -- but what does this suggest about the display order? When entering a URI to many browsers, it may be possible to provide only the domain name and leave the "http://" to be filled in by default, assuming no tail (an approach that does not work for other protocols). The natural display order for the typed domain name on a right-to-left system is fed.cba. Does this change if a protocol identifier, tail, and the corresponding delimiters are specified?

While logic, precedent, and reality suggest that these are questions for user interface design, not IETF protocol specifications, 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, and heuristics that sometimes fail, if each implementation of each application makes its own decisions on these issues.

It should be obvious that any revision of IDNA must be more clear about the distinction between network and display order for complete (fully-qualified) domain names, as well as simply for individual labels, than the original specification was. It is likely that some strong suggestions should be made about display order as well.

6.2. Entry and Display in Applications

Applications can accept domain names using any character set or sets desired by the application developer, and can display domain names in any charset. That is, the IDNA protocol does not 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 and user input of A-labels, but are not encouraged to do so except as an interface for special purposes, possibly for debugging, or to cope with display limitations. A-labels are opaque and ugly, and, where possible, should thus only be exposed to users who absolutely need them. Because IDN labels can be rendered either as the A-labels or U-labels, the application may reasonably have an option for the user to select the preferred method of display; if it does, 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 and the RFC 2822 body parts of SMTP, and 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.

In any place where a protocol or document format allows transmission of the characters in internationalized labels, labels SHOULD be transmitted using whatever character encoding and escape mechanism the protocol or document format uses at that place.

All protocols that use domain name slots already have the capacity for handling domain names in the ASCII charset. Thus, A-labels can inherently be handled by those protocols.

6.3. The Ligature and Digraph Problem

There are a number of languages written with alphabetic scripts in which single phonemes are written using two characters, termed a "digraph", for example, the "ph" in "pharmacy" and "telephone". (Note that characters paired in this manner can also appear consecutively without forming a digraph, as in "tophat".) Certain digraphs are normally 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 (strictly designating setting totally without intervening white space, although the term is sometimes applied to close set pairs). An example of this may be seen when the word "encyclopaedia" is set with a U+00E6 LATIN SMALL LIGATURE AE (and some would not consider that word correctly spelled unless the ligature form was used or the "a" was dropped entirely).

Difficulties arise from the fact that a given ligature may be a completely optional typographic convenience for representing a digraph in one language (as in the above example with some spelling conventions), while in another language it is a single character that may not always be correctly representable by a two-letter sequence

(as in the above example with different spelling conventions). This 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. 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, in parallel to the "umlauted a", it cannot be correctly represented as "oe" and in the Norwegian alphabet, where it is represented, not as "umlauted o", but as "slashed o", U+00F8.

Additional cases with alphabets written right-to-left are described in Section 6.4. This constitutes a problem that cannot be resolved solely by operating on scripts. It is, however, a key concern in the IDN context. Its satisfactory resolution will require support in policies set by registries, which therefore need to be particularly mindful not just of this specific issue, but of all other related matters that cannot be dealt with on an exclusively algorithmic basis.

Just as with the examples of different-looking characters that may be assumed to be the same, it is in general impossible to deal with these situations in a system such as IDNA -- or with Unicode normalization generally -- since determining what to do requires information about the language being used, context, or both. Consequently, these specifications make no attempt to treat these combined characters in any special way. However, their existence provides a prime example of a situation in which 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] to reduce the opportunities for user confusion and fraud that would result from the related strings being registered to different parties.

6.4. 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, may not include any characters with strong left-to-right properties (which excludes other alphabetic characters but permits European digits), and rejects any other string that contains a right-to-left character. This is one of the few places where the IDNA algorithms (both old and new) are required to look at an entire label, not just at individual characters. Unfortunately, the algorithmic model used in IDNA2003 fails when the final character in a right-to-left string requires a combining mark in order to be correctly represented. The mark will be the final code point in the string but is not identified with the right-to-left character attribute and Stringprep therefore rejects the string.

This problem manifests itself in languages written with consonantal alphabets to which diacritical vocalic systems are applied, and in 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). Other languages are still being investigated, but the new rules for right to left scripts are described in [IDNA200X-Bidi].

7. IDNs and the Robustness Principle

The model of IDNs described in this document can be seen as a particular instance of the "Robustness Principle" that has been so important to other aspects of Internet protocol design. This principle is often stated as "Be conservative about what you send and liberal in what you accept" (See, e.g., RFC 1123, Section 1.2.2 [RFC1123]). For IDNs to work well, registries must have or require sensible policies about what is registered -- conservative policies -- and implement and enforce them. Registries, registrars, or other actors who do not do so, or who get too liberal, too greedy, or too weird may deserve punishment that will primarily be meted out in the marketplace or by consumer protection rules and legislation. One can debate whether or not "punishment by browser vendor" is an effective marketplace tool, but it falls into the general category of approaches being discussed here. In any event, the Protocol Police (an important, although mythical, Internet mechanism for enforcing protocol conformance) are going to be worth about as much here as they usually are -- i.e., very little -- simply because, unlike the

marketplace and legal and regulatory mechanisms, they have no enforcement power.

Conversely, resolvers can (and SHOULD or maybe MUST) 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.

As mentioned above, if a string doesn't resolve, it makes no difference whether it simply wasn't registered or was prohibited by some rule.

If resolvers, as a user interface (UI) matter, decide to warn about some strings that are valid under the global rules but that they perceive as dangerous, that is their prerogative and we can only hope that the market (and maybe regulators) will reward the good choices and punish the bad ones. In this context, a resolver that decides a string that is valid under the protocol is dangerous and refuses to look it up is in violation of the protocols (if they are properly defined); one that is willing to look something up, but warns against it, is exercising a UI choice.

8. Migration and Version Synchronization

8.1. Design Criteria

As mentioned above and in RFC 4690, two key goals of this work are to enable applications to be agnostic about whether they are being run in environments supporting any Unicode version from 3.2 onward and to permit incrementally adding permitted scripts and other character collections without disruption. The mechanisms that support this are outlined above, but this section reviews them in a context that may be more helpful to those who need to understand the approach and make plans for it.

1. The general criteria for a putative label, and the collection of characters that make it up, to be considered IDNA-valid are:

   * The characters are "letters", numerals, or otherwise used to write words in some language. Symbols, drawing characters, and various notational characters are permanently excluded -- some because they are actively dangerous in URI, IRI, or

similar contexts and others because there is no evidence that they are important enough to Internet operations or internationalization to justify large numbers of special cases and character-specific handling (additional discussion and rationale for the symbol decision appears in Section 8.5). If strings are read out loud, rather than seen on paper, there are opportunities for considerable confusion between the name of a symbol (and a single symbol may have multiple names) and the symbol itself. 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 as well. 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 not prohibited if the conventions and orthography of a particular language cause that to be possible).

* Characters that are unassigned in the version of Unicode being used by the registry or application are not permitted, even on resolution (lookup). This is because, unlike the conditions contemplated in IDNA2003 (except for right-to-left text), we now understand that tests involving the context of characters (e.g., some characters being permitted only adjacent to other ones of specific types) and integrity tests on complete labels will be needed. Unassigned code points cannot be permitted because one cannot determine the contextual rules that particular code points will require before characters are assigned to them and the properties of those characters fully understood.

* Any character that is mapped to another character by Nameprep2003 or by a current version of NFKC is prohibited as input to IDNA (for either registration or resolution). Implementers of user interfaces to applications are free to make those conversions when they consider them suitable for their operating system environments, context, or users.

Tables used to identify the characters that are IDNA-valid are expected to be driven by the principles above. The principles are not just an interpretation of the tables.

2. For registration purposes, the collection of IDNA-valid characters will be a growing list. The conditions for entry to the list for a set of characters are (i) that they meet the conditions for IDNA-valid characters discussed immediately above and (ii) that consensus can be reached about usage and contextual rules. Because it is likely that such consensus cannot be reached immediately about the correct contextual rules for some

characters -- e.g., the use of invisible ("zero-width") characters to modify presentation forms -- some sets of characters may be deferred from the IDNA-valid set even if they appear in a current version of Unicode. Of course, characters first assigned code points in later versions of Unicode would need to be introduced into IDNA only after those code points are assigned.

3. Anyone entering a label into a DNS zone must properly validate that label -- i.e., be sure that the criteria for an A-label are met -- in order for Unicode version-independence to be possible. In particular:

   * Any label that contains hyphens as its third and fourth characters MUST be IDNA-valid. This implies that, (i) if the third and fourth characters are hyphens, the first and second ones MUST be "xn" until and unless this specification is updated to permit other prefixes and (ii) labels starting in "xn--" MUST be valid A-labels, as discussed in Section 3 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 described above), 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.

   Systems looking up or resolving DNS labels MUST be able to assume that those rules were followed.

4. Anyone looking up a label in a DNS zone MUST

   * Maintain a consistent set of tables, as discussed above. As with registration, the tables need not reflect the latest version of Unicode but they MUST be consistent.

   * Validate labels to be looked up only to the extent of determining that the U-label does not contain either code points prohibited by IDNA (categorized as "NEVER") or code points that are unassigned in its version of Unicode. No attempt should be made to validate contextual rules about characters, including mixed-script label prohibitions, although such rules MAY be used to influence presentation decisions in the user interface.

   By avoiding applying its own interpretation of which labels are

valid as a means of rejecting lookup attempts, the resolver application becomes less sensitive to version incompatibilities with the particular zone registry associated with the domain name.

Under this model, a registry (or entity communicating with a registry to accomplish name registrations) will need to update its tables -- both the Unicode-associated tables and the tables of permitted IDN characters -- to enable a new script or other set of new characters. It will not be affected by newer versions of Unicode, or newly-authorized characters, until and unless it wishes to make those registrations. The registration side is also responsible --under the protocol and to registrants and users-- for much more careful checking than is expected of applications systems that look names up, both checking as required by the protocol and checking required by whatever policies it develops for minimizing risks due to confusable characters and sequences and preserving language or script integrity.

An application or client that looks names up in the DNS will be able to resolve any name that is registered, as long as its version of the Unicode-associated tables is sufficiently up-to-date to interpret all of the characters in the label. It SHOULD distinguish, in its messages to users, 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).

8.2. More Flexibility in User Agents

One key philosophical difference between IDNA2003 and this proposal is that the former provided mappings for many characters into others. These mappings were not reversible: the original string could not be recovered from the form stored in the DNS and, probably as a consequence, users became confused about what characters were valid for IDNs and which ones were not. Too many times, the answer to the question "can this character be used in an IDN" was "it depends on exactly what you mean by 'used'".

IDNA200x does not perform these mappings but, instead, prohibits the characters that would be mapped to others. As examples, while mathematical characters based on Latin ones are accepted as input to IDNA2003, they are prohibited in IDNA200x. Similarly, double-width characters and other variations are prohibited as IDNA input.

Since the rules in [IDNA200X-Tables] provide that only strings that are stable under NFKC are valid, if it is convenient for an

application to perform NFKC normalization before lookup, that operation is safe since this will never make the application unable to look up any valid string.

In many cases these prohibitions should have no effect on what the user can type at resolution time: it is perfectly reasonable for systems that support user interfaces at lookup time, to perform some character mapping that is appropriate to the local environment prior to actual invocation of IDNA as part of the Unicode conversions of [IDNA200X-protocol] above. 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 change in 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 attempted to simulate that behavior by performing case-mapping 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 consistent with having Unicode work correctly (e.g., NFC mapping to resolve different codings for the same character is still necessary) and 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., present in a U-label), then IDNA200x should prohibit upper-case characters as input. Some other considerations reinforce this conclusion. For example, an essential element of the ASCII case-mapping functions is that uppercase(character) must be equal to uppercase(lowercase(character)). That requirement may not be satisfied with IDNs. The relationship between upper case and lower case may even be language-dependent, with different languages (or even the same language in different areas) using different mappings. Of course, the expectations of users who are accustomed to a case-insensitive DNS environment will probably be well-served if user agents perform case mapping prior to IDNA processing, but the IDNA procedures themselves should neither require such mapping nor expect it when it isn't natural to the localized environment.

8.3. The Question of Prefix Changes

The conditions that would require a change in the IDNA "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.

8.3.1. Conditions requiring a prefix change

An IDN prefix change is needed if a given string would resolve or otherwise be interpreted differently depending on the version of the protocol or tables being used. Consequently, work to update IDNs 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 IDNA200x.

2. An input string that is valid under IDNA2003 and also valid under IDNA200x yields two different A-labels with the different versions of IDNA. This condition is believed to be essentially equivalent to the one above.

   Note, however, 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 8.3.2, 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 specific script information in that string, rather than having it be just a string of characters.

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

8.3.2. Conditions not requiring a prefix change

In particular, 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 require a prefix change.

2. Adjustments in Stringprep tables or IDNA actions, including normalization definitions, that do not affect characters that have already been invalid under IDNA2003.

3. Changes in the style of definitions of Stringprep or Nameprep that do not alter the actions performed by them.

8.4. Stringprep Changes and Compatibility

Concerns have been expressed about problems for non-DNS uses of Stringprep being caused by changes to the specification intended to improve the handling of IDNs, most notably as this might affect identification and authentication protocols. Section 8.3, above, essentially also applies in this context. The proposed new inclusion tables [IDNA200X-Tables], the reduction in the number of characters permitted as input for registration or resolution (Section 5), and even the proposed changes in handling of right-to-left strings [IDNA200X-Bidi] either give interpretations to strings prohibited under IDNA2003 or prohibit strings that IDNA2003 permitted. Strings that are valid under both IDNA2003 and IDNA200x, and the corresponding versions of Stringprep, are not changed in interpretation. This protocol does not use either Nameprep or Stringprep as specified in IDNA2003.

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

Perhaps even more important in practice, since most other known uses of Stringprep encode or process characters that are already in normalized form and expect the use of only those characters that can be used in writing words of languages, the changes proposed here and in [IDNA200X-Tables] are unlikely to have any effect at all, especially not on registries and registrations that follow rules already in existence when this work started.

8.5. The Symbol Question

[[anchor37: Move this material and integrate with the Symbol discussion above???]]

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 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 bans the symbols entirely. There are several reasons for this, which include:

- 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 LDH rules --which have served the Internet very well-- to a Unicode base rather than an ASCII base.

- Unicode names for letters are fairly intuitive, recognizable to uses of the relevant script, and unambiguous. Symbol names are more problematic 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 as result, symbols are a very poor basis for reliable communications. 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.

8.6. Other Compatibility Issues

The existing (2003) IDNA model has several odd artifacts which occur largely by accident. 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 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 proposed IDNA200x 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.

9. Acknowledgments

The editor and contributors would like to express their thanks to those who contributed significant early review comments, sometimes accompanied by text, especially Mark Davis, 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 Frank Ellerman, Michael Everson, Asmus Freytag, Michel Suignard, and Ken Whistler, although, as usual, they bear little or no responsibility for the conclusions the editor and contributors reached after receiving their suggestions. Thanks are also due to Vint Cerf, Debbie Garside, and Jefsey Morphin for conversations that led to considerable improvements in the content of this document.

10. Contributors

While the listed editor held the pen, this document 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. 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.

11. IANA Considerations

11.1. IDNA Permitted Character Registry

The distinction between "MAYBE" code points and those classified into "ALWAYS" and "NEVER" (see Section 5) requires a registry of characters and scripts and their categories. IANA is requested to establish that registry, using the "expert reviewer" model. Unlike usual practice, we recommend that the "expert reviewer" be a committee that reflects expertise on the relevant scripts, and encourage IANA, the IESG, and IAB to establish liaisons and work together with other relevant standards bodies to populate that committee and its procedures over the long term.

11.2. IDNA Context Registry

For characters that are defined in the permitted character as requiring a contextual rule, IANA will create and maintain a list of approved contextual rules, using the registration methods described above. IANA should develop a format for that registry, or a copy of it maintained in parallel, that is convenient for retrieval and

machine processing and publish the location of that version.

11.3. IANA Repository of TLD IDN Practices

This registry is maintained by IANA at the request of ICANN, 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 effect on that registry and no IANA action is required as a result.

12. Security Considerations

Security on the Internet partly relies on the DNS. Thus, any change to the characteristics of the DNS can change the security of much of the Internet.

Domain names are used by users to identify and connect to Internet servers. The security of the Internet is compromised if a user entering a single internationalized name is connected to different servers based on different interpretations of the internationalized domain name.

When systems use local character sets other than ASCII and Unicode, this specification leaves the the problem of transcoding between the local character set and Unicode up to the application or local system. If different applications (or different versions of one application) implement different transcoding rules, they could interpret the same name differently and contact different servers. This problem is not solved by security protocols like TLS that do not take local character sets into account.

To help prevent confusion between characters that are visually similar, it is suggested that implementations provide visual indications where a domain name contains multiple scripts. Such mechanisms can also be used to show when a name contains a mixture of simplified and traditional Chinese characters, or to distinguish zero and one from O and l. DNS zone adminstrators may impose restrictions (subject to the limitations identified elsewhere in this document) that try to minimize characters that have similar appearance or similar interpretations. It is worth noting that there are no comprehensive technical solutions to the problems of confusable characters. One can reduce the extent of the problems in various ways, but probably never eliminate it. Some specific suggestion about identification and handling of confusable characters appear in a Unicode Consortium publication [???]

The registration and resolution models described above and in

[IDNA200X-protocol] change the mechanisms available for applications and resolvers to determine the validity of labels they encounter. In some respects, the ability to test is strengthened. For example, putative labels that contain unassigned code points will now be rejected, while IDNA2003 permitted them (something that is now recognized as a considerable source of risk). On the other hand, the protocol specification no longer assumes that the application that looks up a name will be able to determine, and apply, information about the protocol version used in registration. In theory, that may increase risk since the application will be able to do less pre-lookup validation. In practice, the protection afforded by that test has been largely illusory for reasons explained in RFC 4690 and above.

Any change to Stringprep or, more broadly, the IETF's model of the use of internationalized character strings in different protocols, creates some risk of inadvertent changes to those protocols, invalidating deployed applications or databases, and so on. Our current hypothesis is that the same considerations that would require changing the IDN prefix (see Section 8.3.2) are the ones that would, e.g., invalidate certificates or hashes that depend on Stringprep, but those cases require careful consideration and evaluation. More important, it is not necessary to change Stringprep2003 at all in order to make the IDNA changes contemplated here. It is far preferable to create a separate document, or separate profile components, for IDN work, leaving the question of upgrading to other protocols to experts on them and eliminating any possible synchronization dependency between IDNA changes and possible upgrades to security protocols or conventions.

13. Change Log

[[anchor44: RFC Editor: Please remove this section.]]

13.1. Version -01

Version -01 of this document is a considerable rewrite from -00. Many sections have been clarified or extended and several new sections have been added to reflect discussions in a number of contexts since -00 was issued.

13.2. Version -02

- Corrected several editorial errors including an accidentally-introduced misstatement about NFKC.

- Extensively revised the document to synchronize its terminology with version 03 of [IDNA200X-Tables] and to provide a better conceptual framework for its categories and how they are used. Added new material to clarify terminology and relationships with other efforts. More subtle changes in this version lay the groundwork for separating the document into a conceptual overview and a protocol specification for version 03.

13.3. Version -03

- Removed protocol materials to a separate document and incorporated rationale and explanation materials from the original specification in RFC 3960 into this document. Cleaned up earlier text to reflect a more mature specification and restructured several sections and added additional rationale material.

- Strengthened and clarified the A-label / U-label/ LDH-label definition.

- Retitled the document to reflect its evolving role.

13.4. Version -04

- Moved more text from "protocol" and further reorganized material.

- Provided new material on "Contextual Rule Required.

- Improved consistency of terminology, both internally and with the "tables" document.

- Improved the IANA Considerations section and discussed the existing IDNA-related registry.

- More small changes to increase consistency.

13.5. Version -05

Changed "YES" category back to "ALWAYS" to re-synch with the tables document and provide clearer terminology.

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.

[IDNA200X-Bidi]
Alvestrand, H. and C. Karp, "An IDNA problem in right-to-left scripts", July 2007, <http://www.ietf.org/internet-drafts/draft-alvestrand-idna-bidi-01.txt>.

[IDNA200X-Tables]
Faltstrom, P., "The Unicode Codepoints and IDN", November 2007, <http://stupid.domain.name/idnabis/draft-faltstrom-idnabis-tables-03.txt>.

A version of this document, is available in HTML format at http://stupid.domain.name/idnabis/draft-faltstrom-idnabis-tables-03.txt

[IDNA200X-protocol]
Klensin, J., "Internationalizing Domain Names in Applications (IDNA): Protocol", November 2007, <http://www.ietf.org/internet-drafts/draft-klensin-idnabis-protocol-01.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3454] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.

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

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

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

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

[Unicode-UAX15] The Unicode Consortium, "Unicode Standard Annex #15: Unicode Normalization Forms", 2006, <http://www.unicode.org/reports/tr15/>.

[Unicode32] The Unicode Consortium, "The Unicode Standard, Version 3.0", 2000.

(Reading, MA, Addison-Wesley, 2000. ISBN 0-201-61633-5). Version 3.2 consists of the definition in that book as amended by the Unicode Standard Annex #27: Unicode 3.1 (http://www.unicode.org/reports/tr27/) and by the Unicode Standard Annex #28: Unicode 3.2 (http://www.unicode.org/reports/tr28/).

[Unicode40] The Unicode Consortium, "The Unicode Standard, Version 4.0", 2003.

[Unicode50] The Unicode Consortium, "The Unicode Standard, Version 5.0", 2007.

Boston, MA, USA: Addison-Wesley. ISBN 0-321-48091-0

14.2. Informative References

[ICANN-Guidelines] ICANN, "IDN Implementation Guidelines", 2006, <http://www.icann.org/topics/idn/>.

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

[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.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

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

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

Author's Address

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

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

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).