Network Working Group
Request for Comments: 4690
Category: Informational

J. Klensin
P. Faltstrom
Cisco Systems
C. Karp
Swedish Museum of Natural History
IAB

September 2006

Review and Recommendations for Internationalized Domain Names (IDNs)

Status of This Memo

Copyright Notice

Abstract

This note describes issues raised by the deployment and use of Internationalized Domain Names. It describes problems both at the time of registration and for use of those names in the DNS. It recommends that IETF should update the RFCs relating to IDNs and a framework to be followed in doing so, as well as summarizing and identifying some work that is required outside the IETF. In particular, it proposes that some changes be investigated for the Internationalizing Domain Names in Applications (IDNA) standard and its supporting tables, based on experience gained since those standards were completed.

Table of Contents

1. Introduction

1.1. The Role of IDNs and This Document

While IDNs have been advocated as the solution to a wide range of problems, this document is written from the perspective that they are no more and no less than DNS names, reflecting the same requirements for use, stability, and accuracy as traditional "hostnames", but using a much larger collection of permitted characters. In particular, while IDNs represent a step toward an Internet that is equally accessible from all languages and scripts, they, at best, address only a small part of that very broad objective. There has been controversy since IDNs were first suggested about how important they will actually turn out to be; that controversy will probably continue. Accessibility from all languages is an important objective, hence it is important that our standards and definitions for IDNs be smoothly adaptable to additional scripts as they are added to the Unicode character set.

The utility of IDNs must be evaluated in terms of their application by users and in protocols: the ability to simply put a name into the DNS and retrieve it is not, in and of itself, important. From this point of view, IDNs will be useful and effective if they provide stable and predictable references -- references that are no less stable and predictable, and no less secure, than their ASCII counterparts.

This combination of objectives and criteria has proven very difficult to satisfy. Experience in developing the IDNA standard and during the initial years of its implementation and deployment suggests that it may be impossible to fully satisfy all of them and that engineering compromises are needed to yield a result that is workable, even if not completely satisfactory. Based on that experience and issues that have been raised, it is now appropriate to review some of the implications of IDNs, the decisions made in defining them, and the foundation on which they rest and determine whether changes are needed and, if so, which ones.

The design of the DNS itself imposes some additional constraints. If the DNS is to remain globally interoperable, there are specific characteristics that no implementation of IDNs, or the DNS more generally, can change. For example, because the DNS is a global hierarchal administrative namespace with only a single name at any given node, there is one and only one owner of each domain name. Also, when strings are looked up in the DNS, positive responses can only reflect exact matches: if there is no exact match, then one gets an error reply, not a list of near matches or other supplemental information. Searches and approximate matchings are not possible.

Finally, because the DNS is a distributed system where any server might cache responses, and later use those cached responses to attempt to satisfy queries before a global lookup is done, every server must use the same matching criteria.

1.2. Status of This Document and Its Recommendations

This document reviews the IDN landscape from an IETF perspective and presents the recommendations and conclusions of the IAB, based partially on input from an ad hoc committee charged with reviewing IDN issues and the path forward (see Section 7). Its recommendations are advice to the IETF, or in a few cases to other bodies, for topics to be investigated and actions to be taken if those bodies, after their examinations, consider those actions appropriate.

1.3. The IDNA Standard

During 2002, the IETF completed the following RFCs that, together, define IDNs:

RFC 3454 Preparation of Internationalized Strings ("Stringprep") [RFC3454].
Stringprep is a generic mechanism for taking a Unicode string and converting it into a canonical format. Stringprep itself is just a collection of rules, tables, and operations. Any protocol or algorithm that uses it must define a "Stringprep profile", which specifies which of those rules are applied, how, and with which characteristics.

RFC 3490 Internationalizing Domain Names in Applications (IDNA) [RFC3490].
IDNA is the base specification in this group. It specifies that Nameprep is used as the Stringprep profile for domain names, and that Punycode is the relevant encoding mechanism for use in generating an ASCII-compatible ("ACE") form of the name. It also applies some additional conversions and character filtering that are not part of Nameprep.

RFC 3491 Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN) [RFC3491].
Nameprep is designed to meet the specific needs of IDNs and, in particular, to support case-folding for scripts that support what are traditionally known as upper- and lowercase forms of the same letters. The result of the Nameprep algorithm is a string containing a subset of the Unicode Character set, normalized and case-folded so that case-insensitive comparison can be made.

RFC 3492 Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA) [RFC3492]. Punycode is a mechanism for encoding a Unicode string in ASCII characters. The characters used are the same the subset of characters that are allowed in the hostname definition of DNS, i.e., the "letter, digit, and hyphen" characters, sometimes known as "LDH".

1.4. Unicode Documents

Unicode is used as the base, and defining, character set for IDNs. Unicode is standardized by the Unicode Consortium, and synchronized with ISO to create ISO/IEC 10646 [ISO10646]. At the time the RFCs mentioned earlier were created, Unicode was at Version 3.2. For reasons explained later, it was necessary to pick a particular, then-current, version of Unicode when IDNA was adopted. Consequently, the RFCs are explicitly dependent on Unicode Version 3.2 [Unicode32]. There is, at present, no established mechanism for modifying the IDNA RFCs to use newer Unicode versions (see Section 3.1).

Unicode is a very large and complex character set. (The term "character set" or "charset" is used in a way that is peculiar to the IETF and may not be the same as the usage in other bodies and contexts.) The Unicode Standard and related documents are created and maintained by the Unicode Technical Committee (UTC), one of the committees of the Unicode Consortium.

The Consortium first published The Unicode Standard [Unicode10] in 1991, and continues to develop standards based on that original work. Unicode is developed in conjunction with the International Organization for Standardization, and it shares its character repertoire with ISO/IEC 10646. Unicode and ISO/IEC 10646 function equivalently as character encodings, but The Unicode Standard contains much more information for implementers, covering -- in depth -- topics such as bitwise encoding, collation, and rendering. The Unicode Standard enumerates a multitude of character properties, including those needed for supporting bidirectional text. The Unicode Consortium and ISO standards do use slightly different terminology.

1.5. Definitions

The following terms and their meanings are critical to understanding the rest of this document and to discussions of IDNs more generally. These terms are derived from [RFC3536], which contains additional discussion of some of them.

1.5.1. Language

A language is a way that humans interact. The use of language occurs in many forms, including speech, writing, and signing.

Some languages have a close relationship between the written and spoken forms, while others have a looser relationship. RFC 3066 [RFC3066] discusses languages in more detail and provides identifiers for languages for use in Internet protocols. Computer languages are explicitly excluded from this definition. The most recent IETF work in this area, and on script identification (see below), is documented in [RFC4645] and [RFC4646].

1.5.2. Script

A script is a set of graphic characters used for the written form of one or more languages. This definition is the one used in [ISO10646].

Examples of scripts are Arabic, Cyrillic, Greek, Han (the so-called ideographs used in writing Chinese, Japanese, and Korean), and "Latin". Arabic, Greek, and Latin are, of course, also names of languages.

Historically, the script that is known as "Latin" in Unicode and most contexts associated with information technology standards is known in the linguistic community as "Roman" or "Roman-derived". The latter terminology distinguishes between the Latin language and the characters used to write it, especially in Republican times, from the much richer and more decorated script derived and adapted from those characters. Since IDNA is defined using Unicode and that standard used the term "LATIN" in its character names and descriptions, that terminology will be used in this document as well except when "Roman-derived" is needed for clarity. However, readers approaching this document from a cultural or linguistic standpoint should be aware that the use of, or references to, "Latin script" in this document refers to the entire collection of Roman-derived characters, not just the characters used to write the Latin language. Some other issues with script identification and relationships with other standards are discussed in [RFC4646].

1.5.3. Multilingual

The term "multilingual" has many widely-varying definitions and thus is not recommended for use in standards. Some of the definitions relate to the ability to handle international characters; other definitions relate to the ability to handle multiple charsets; and still others relate to the ability to handle multiple languages.

While this term has been deprecated for IETF-related uses and does not otherwise appear in this document, a discussion here seemed appropriate since the term is still widely used in some discussions of IDNs.

1.5.4. Localization

Localization is the process of adapting an internationalized application platform or application to a specific cultural environment. In localization, the same semantics are preserved while the syntax or presentation forms may be changed.

Localization is the act of tailoring an application for a different language or script or culture. Some internationalized applications can handle a wide variety of languages. Typical users understand only a small number of languages, so the program must be tailored to interact with users in just the languages they know.

Somewhat different definitions for localization and internationalization (see below) are used by groups other than the IETF. See [W3C-Localization] for one example.

1.5.5. Internationalization

In the IETF, the term "internationalization" is used to describe adding or improving the handling of non-ASCII text in a protocol. Other bodies use the term in other ways, often with subtle variation in meaning. The term "internationalization" is often abbreviated "i18n" (and localization as "l10n").

Many protocols that handle text only handle the characters associated with one script (often, a subset of the characters used in writing English text), or leave the question of what character set is used up to local guesswork (which leads to interoperability problems). Adding non-ASCII text to such a protocol allows the protocol to handle more scripts, with the intention of being able to include all of the scripts that are useful in the world. It is naive (sic) to believe that all English words can be written in ASCII, various mythologies notwithstanding.

1.6. Statements and Guidelines

When the IDNA RFCs were published, the IESG and ICANN made statements that were intended to guide deployment and future work. In recent months, ICANN has updated its statement and others have also made contributions. It is worth noting that the quality of understanding of internationalization issues as applied to the DNS has evolved

considerably over the last few years. Organizations that took specific positions a year or more ago might not make exactly the same statements today.

1.6.1. IESG Statement

The IESG made a statement on IDNA [IESG-IDN]:

> IDNA, through its requirement of Nameprep [RFC3491], uses equivalence tables that are based only on the characters themselves; no attention is paid to the intended language (if any) for the domain name. However, for many domain names, the intended language of one or more parts of the domain name actually does matter to the users.
>
> Similarly, many names cannot be presented and used without ambiguity unless the scripts to which their characters belong are known. In both cases, this additional information should be of concern to the registry.

The statement is longer than this, but these paragraphs are the important ones. The rest of the statement consists of explanations and examples.

1.6.2. ICANN Statements

1.6.2.1. Initial ICANN Guidelines

Soon after the IDNA standards were adopted, ICANN produced an initial version of its "IDN Guidelines" [ICANNv1]. This document was intended to serve two purposes. The first was to provide a basis for releasing the Generic Top Level Domain (gTLD) registries that had been established by ICANN from a contractual restriction on the registration of labels containing hyphens in the third and fourth positions. The second was to provide a general framework for the development of registry policies for the implementation of IDNs.

One of the key components of this framework prescribed strict compliance with RFCs 3490, 3491, and 3492. With the framework, ICANN specified that IDNA was to be the sole mechanism to be used in the DNS to represent IDNs.

Limitations on the characters available for inclusion in IDNs were mandated by two mechanisms. The first was by requiring an "inclusion-based approach (meaning that code points that are not explicitly permitted by the registry are prohibited) for identifying permissible

code points from among the full Unicode repertoire." The second mechanism required the association of every IDN with a specific language, with additional policies also being language based:

"In implementing the IDN standards, top-level domain registries will
(a) associate each registered internationalized domain name with one language or set of languages,
(b) employ language-specific registration and administration rules that are documented and publicly available, such as the reservation of all domain names with equivalent character variants in the languages associated with the registered domain name, and,
(c) where the registry finds that the registration and administration rules for a given language would benefit from a character variants table, allow registrations in that language only when an appropriate table is available. ... In implementing the IDN standards, top-level domain registries should, at least initially, limit any given domain label (such as a second-level domain name) to the characters associated with one language or set of languages only."

It was left to each TLD registry to define the character repertoire it would associate with any given language. This led to significant variation from registry to registry, with further heterogeneity in the underlying language-based IDN policies. If the guidelines had made provision for IDN policies also being based on script, a substantial amount of the resulting ambiguity could have been avoided. However, they did not, and the sequence of events leading to the present review of IDNA was thus triggered.

1.6.2.2. ICANN Version 2 Guidelines

One of the responses of the TLD registries to what was widely perceived as a crisis situation was to invoke the mechanism described in the initial guidelines: "As the deployment of IDNs proceeds, ICANN and the IDN registries will review these Guidelines at regular intervals, and revise them as necessary based on experience."

The pivotal requirement was the modification of the guidelines to permit script-based policies for IDNs. Further concern was expressed about the need for realistically implementable mechanisms for the propagation of TLD registry policies into the lower levels of their name trees. In addition to the anticipated increase of constraint on the protocol level, one obvious additional approach would be to replace the guidelines by an instrument that itself had clear status in the IETF's normative framework. A BCP was therefore seen as the appropriate focus for longer-term effort. The most pressing issues would be dealt with in the interim by incremental modification to the guidelines, but no need was seen for the detailed further development of those guidelines once that incremental modification was complete.

The outcome of this action was a version 2.0 of the guidelines [ICANNv2], which was endorsed by the ICANN Board on November 8, 2005 for a period of nine months. The Board stated further that it "tasks the IDN working group to continue its important work and return to the board with specific IDN improvement recommendations before the ICANN Meeting in Morocco" and "supports the working group's continued action to reframe the guidelines completely in a manner appropriate for further development as a Best Current Practices (BCP) document, to ensure that the Guideline directions will be used deeper into the DNS hierarchy and within TLD's where ICANN has a lesser policy relationship."

Retaining the inclusion-based approach established in version 1.0, the crucial addition to the policy framework is that:

"All code points in a single label will be taken from the same script as determined by the Unicode Standard Annex #24: Script Names at http://www.unicode.org/reports/tr24. Exception to this is permissible for languages with established orthographies and conventions that require the commingled use of multiple scripts. In such cases, visually confusable characters from different scripts will not be allowed to coexist in a single set of permissible codepoints unless a corresponding policy and character table is clearly defined."

Additionally:

"Permissible code points will not include: (a) line symbol-drawing characters (as those in the Unicode Box Drawing block), (b) symbols and icons that are neither alphanumeric nor ideographic language characters, such as typographic and pictographic dingbats, (c) characters with well-established functions as protocol elements, (d) punctuation marks used solely to indicate the structure of sentences."

Attention has been called to several points that are not adequately dealt with (if at all) in the version 2.0 guidelines but that ought to be included in the policy framework without waiting for the production and release of a document based on a "best practices" model. The term "BCP" above does not necessarily refer to an IETF consensus document.

The intention in November 2005 was for the recommended major revision to be put to the ICANN Board prior to its meeting in Morocco (in late June 2006), but for the changes to be collated incrementally and appear in interim version 2.n releases of the guidelines. The IAB's understanding is that, while there has been some progress with this,

other issues relating to IDNs subsequently diverted much of the energy that was intended to be devoted to the more extensive treatment of the guidelines.

2. General Problems and Issues

This section interweaves problems and issues of several types. Each subsection outlines something that is perceived to be a problem or issue "with IDNs", therefore needing correction. Some of these issues can be at least partially resolved by making changes to elements of the IDNA protocol or tables. Others will exist as long as people have expectations of IDNs that are inconsistent with the basic DNS architecture. It is important to identify this entire range of problems because users, registrants, and policy makers often do not understand the protocol and other technical issues but only the difference between what they believe happens or should happen and what actually happens. As long as those differences exist, there will be demands for functionality or policy changes for IDNs. Of course, some of these demands will be less realistic than others, but even the realistic ones should be understood in the same context as the others.

Most of the issues that have been raised, and that are discussed in this document, exist whether IDNA remains tied to Unicode 3.2 or whether migration to new Unicode versions is contemplated. A migration path is necessary to accommodate newly-coded scripts and to permit the maximum number of languages and scripts to be represented in domain names. However, the migration issues are largely separate from those involving a single Unicode version or Version 3.2 in particular, so they have been separated into this section and Section 3.

2.1. User Conceptions, Local Character Sets, and Input issues

The labels of the DNS are just strings of characters that are not inherently tied to a particular language. As mentioned briefly in the Introduction, DNS labels that could not lexically be words in any language are possible and indeed common. There appears to be no reason to impose protocol restrictions on IDNs that would restrict them more than all-ASCII hostname labels have been restricted. For that reason, even describing DNS labels or strings of them as "names" is something of a misnomer, one that has probably added to user confusion about what to expect.

Ordinarily, people use "words" when they think of things and wish others to think of them too, for example, "orange", "tree", "restaurant" or "Acme Inc". Words are normally in a specific language, such as English or Swedish. The character-string labels

supported by the DNS are, as suggested above, not inherently "words". While it is useful, especially for mnemonic value or to identify objects, for actual words to be used as DNS labels, other constraints on the DNS make it impossible to guarantee that it will be possible to represent every word in every language as a DNS label, internationalized or not.

When writing or typing the label (or word), a script must be selected and a charset must be picked for use with that script. The choice of charset is typically not under the control of the user on a per-word or per-document basis, but may depend on local input devices, keyboard or terminal drivers, or other decisions made by operating system or even hardware designers and implementers.

If that charset, or the local charset being used by the relevant operating system or application software, is not Unicode, a further conversion must be performed to produce Unicode. How often this is an issue depends on estimates of how widely Unicode is deployed as the native character set for hardware, operating systems, and applications. Those estimates differ widely, but it should be noted that, among other difficulties:

- ISO 8859 versions [ISO.8859.2003] and even national variations of ISO 646 [ISO.646.1991], are still widely used in parts of Europe;

- code-table switching methods, typically based on the techniques of ISO 2022 [ISO.2022.1986] are still in general use in many parts of the world, especially in Japan with Shift-JIS and its variations; and

- computing, systems, and communications in China tend to use one or more of the national "GB" standards rather than native Unicode.

Additionally, not all charsets define their characters in the same way and not all preexisting coding systems were incorporated into Unicode without changes. Sometimes local distinctions were made that Unicode does not make or vice versa. Consequently, conversion from other systems to Unicode may potentially lose information.

The Unicode string that results from this processing -- processing that is trivial in a Unicode-native system but that may be significant in others -- is then used as input to IDNA.

2.2. Examples of Issues

While much of the discussion below is stated in terms of Unicode codings and associated rules, the IAB believes that some of the issues are actually not about the Unicode character set per se, but about how distributed matching systems operate in reality, and about what implications the distributed delayed search for stored data that characterizes the DNS has on the mapping algorithms.

2.2.1. Language-Specific Character Matching

There are similar words that can be expressed in multiple languages. Consider, for example, the name Torbjorn in Norwegian and Swedish. In Norwegian it is spelled with the character U+00F8 (LATIN SMALL LETTER O WITH STROKE) in the second syllable, while in Swedish it is spelled with U+00F6 (LATIN SMALL LETTER O WITH DIAERESIS). Those characters are not treated as equivalent according to the Unicode Standard and its Annexes while most people speaking Swedish, Danish, or Norwegian probably think they are equivalent.

It is neither possible nor desirable to make these characters equivalent on a global basis. To do so would, for this example, rationalize the situation in Sweden while causing considerable confusion in Germany because the U+00F8 character is never used in the German language. But the "variant" model introduced in [RFC3743] and [RFC4290] can be used by a registry to prevent the worst consequence of the possible confusion, by ensuring either that both names are registered to the same party in a given domain or that one of them is completely prohibited.

2.2.2. Multiple Scripts

There are languages in the world that can be expressed using multiple scripts. For example, some Eastern European and Central Asian languages can be expressed in either Cyrillic or Latin (see Section 1.5.2) characters, or some African and Southeast Asian languages can be expressed in either Arabic or Latin characters. A few languages can even be written in three different scripts. In other cases, the language is typically written in a combination of scripts (e.g., Kanji, Kana, and Romaji for Japanese; Hangul and Hanji for Korean). Because of this, the same word, in the same language, can be expressed in different ways. For some languages, only a single script is normally used to write a single word; for others, mixed scripts are required; and, for still others, special circumstances may dictate mixing scripts in labels although that is not normally done for "words". For IDN purposes, these variations make the definition of "script" extremely sensitive, especially since ICANN is now recommending that it be used as the primary basis for

registry policies. However essential it may be to prohibit mixed-script labels, additional policy nuance is required for "languages with established orthographies and conventions that require the commingled use of multiple scripts".

2.2.3. Normalization and Character Mappings

Unicode contains several different models for representing characters. The Chinese (Han)-derived characters of the "CJK" (Chinese, Japanese, and Korean) languages are "unified", i.e., characters with common derivation and similar appearances are assigned to the same code point. European characters derived from a Greek-Latin base are separated into separate code blocks for Latin, Greek, and Cyrillic even when individual characters are identical in both form and semantics. Separate code points based on font differences alone are generally prohibited, but a large number of characters for "mathematical" use have been assigned separate code points even though they differ from base ASCII characters only by font attributes such as "script", "bold", or "italic". Some characters that often appear together are treated as typographical digraphs with specific code points assigned to the combination, others require that the two-character sequences be used, and still others are available in both forms. Some Roman-derived letters that were developed as decorated variations on the basic Latin letter collection (e.g., by addition of diacritical marks) are assigned code points as individual characters, others must be built up as two (or more) character sequences using "combining characters".

Many of these differences result from the desire to maintain backward compatibility while the standard evolved historically, and are hence understandable. However, the DNS requires precise knowledge of which codes and code sequences represent the same character and which ones do not. Limiting the potential difficulties with confusable characters (see Section 2.2.6) requires even more knowledge of which characters might look alike in some fonts but not in others. These variations make it difficult or impossible to apply a single set of rules to all of Unicode and, in doing so, satisfy everyone and their perceived needs. Instead, more or less complex mapping tables, defined on a character-by-character basis, are required to "normalize" different representations of the same character to a single form so that matching is possible.

Unless normalization rules, such as those that underlie Nameprep, are applied, characters that are essentially identical will not match in the DNS, creating many opportunities for problems. The most common of these problems is that, due to the processing applied (and discussed above) before a word is represented as a Unicode string, a single word can end up being expressed as several different Unicode

strings. Even if normalization rules are applied, some strings that are considered identical by users will not compare equal. That problem is discussed in more detail elsewhere in this document, particularly in Section 3.2.1.

IDNA attempts to compensate for these problems by using a normalization algorithm defined by the Unicode Consortium. This algorithm can change a sequence of one or more Unicode characters to another set of characters. One example is that the base character U+0061 (LATIN SMALL LETTER A) followed by U+0308 (COMBINING DIAERESIS) is changed to the single Unicode character U+00E4 (LATIN SMALL LETTER A WITH DIAERESIS).

This Unicode normalization process accounts only for simple character equivalences, not equivalences that are language or script dependent. For example, as mentioned above, the characters U+00F8 (LATIN SMALL LETTER O WITH STROKE) and U+00F6 (LATIN SMALL LETTER O WITH DIAERESIS) are considered to match in Swedish (and some other languages), but not for all languages that use either of the characters. Having these characters be treated as equivalent in some contexts and not in others requires decisions and mechanisms that, in turn, depend much more on context than either IDNA or the Unicode character-based normalization tables can provide.

Additional complications occur if the sequences are more complicated or if an attacker is making a deliberate effort to confuse the normalization process. For example, if the sequence U+0069 U+0307 (LATIN SMALL LETTER I followed by COMBINING DOT ABOVE) appears, the Unicode Normalization Method known as NFKC maps it into U+00EF (LATIN SMALL LETTER I WITH DIAERESIS), which is what one would predict. But consider U+0131 U+0308 (LATIN SMALL LETTER DOTLESS I and COMBINING DIAERESIS): is that the same character? Is U+0131 U+0307 U+0307 (dotless i and two combining dot-above characters) equivalent to U+00EF or U+0069, or neither? NFKC does not appear to tell us, nor does the definition of U+0307 appear to tell us what happens when it is combined with other "symbol above" arrangements (unlike some of the "accent above" combining characters, which more or less specify kerning). Similar issues arise when U+00EF is combined with various dot-above combining characters. Each of these questions provides some opportunities for spoofing if different display implementations interpret the rules in different ways.

If we leave Latin scripts and examine those based on Chinese characters, we see there is also an absence of specific, lexigraphic, rules for transformations between Traditional and Simplified Chinese. Even if there were such rules, unification of Japanese and Korean

characters with Chinese ones would make it impossible to normalize Traditional Chinese into Simplified Chinese ones without causing problems in Japanese and Korean use of the same characters.

More generally, while some mappings, such as those between precomposed Latin script characters and the equivalent multiple code point composed character sequences, depend only on the characters themselves, in many or most cases, such as the case with Swedish above, the mapping is language or culturally dependent. There have been discussions as to whether different canonicalization rules (in addition to or instead of Unicode normalization) should be, or could be, applied differently to different languages or scripts. The fact that most scripts included in Unicode have been initially incorporated by copying an existing standard more or less intact has impact on the optimization of these algorithms and on forward compatibility. Even if the language is known and language-specific rules can be defined, dependencies on the language do not disappear. Canonicalization operations are not possible unless they either depend only on short sequences of text or have significant context available that is not obvious from the text itself. DNS lookups and many other operations do not have a way to capture and utilize the language or other information that would be needed to provide that context.

These variations in languages and in user perceptions of characters make it difficult or impossible to provide uniform algorithms for matching Unicode strings in a way that no end users are ever surprised by the result. For closely-related scripts or characters, surprises may even be frequent. However, because uniform algorithms are required for mappings that are applied when names are looked up in the DNS, the rules that are chosen will always represent an approximation that will be more or less successful in minimizing those user surprises. The current Nameprep and Stringprep algorithms use mapping tables to "normalize" different representations of the same text to a single form so that matching is possible.

More details on the creation of the normalization algorithms can be found in the Unicode Specification and the associated Technical Reports [UTR] and Annexes. Technical Report #36 [UTR36] and [UTR39] are specifically related to the IDN discussion.

2.2.4. URLs in Printed Form

URLs and other identifiers appear, not only in electronic forms from which they can (at least in principle) be accurately copied and "pasted" but in printed forms from which the user must transcribe them into the computer system. This is often known as the "side-of-the-bus problem" because a particularly problematic version of it

requires that the user be able to observe and accurately remember a URL that is quickly glimpsed in a transient form -- a billboard seen while driving, a sign on the side of a passing vehicle, a television advertisement that is not frequently repeated or on-screen for a long time, and so on.

The difficulty, in short, is that two Unicode strings that are actually different might look exactly the same, especially when there is no time to study them. This is because, for example, some glyphs in Cyrillic, Greek, and Latin do look the same, but have been assigned different code points in Unicode. Worse, one needs to be reasonably familiar with a script and how it is used to understand how much characters can reasonably vary as the result of artistic fonts and typography. For example, there are a few fonts for Latin characters that are sufficiently highly ornamented that an observer might easily confuse some of the characters with characters in Thai script. Uppercase ITC Blackadder (a registered trademark of International Typeface Corporation) and Curlz MT are two fairly obvious examples; these fonts use loops at the end of serifs, creating a resemblance to Thai (in some fonts) for some characters.

2.2.5. Bidirectional Text

Some scripts (and because of that some words in some languages) are written not left to right, but right to left. And, to complicate things, one might have something written in Arabic script right to left that includes some characters that are read from left to right, such as European-style digits. This implies that some texts might have a mixed left-to-right AND right-to-left order (even though in most implementations, and in IDNA, all texts have a major direction, with the other as an exception).

IDNA permits the inclusion of European digits in a label that is otherwise a sequence of right-to-left characters, but prohibits most other mixed-directional (or bidirectional) strings. This prohibition can cause other problems such as the rejection of some otherwise linguistically and culturally sensible strings. As Unicode and conventions for handling so-called bidirectional ("BIDI") strings evolve, the prohibition in IDNA should be reviewed and reevaluated.

2.2.6. Confusable Character Issues

Similar-looking characters in identifiers can cause actual problems on the Internet since they can result, deliberately or accidentally, in people being directed to the wrong host or mailbox by believing that they are typing, or clicking on, intended characters that are different from those that actually appear in the domain name or reference. See Section 4.1.3 for further discussion of this issue.

IDNs complicate these issues, not only by providing many additional characters that look sufficiently alike to be potentially confused, but also by raising new policy questions. For example, if a language can be written in two different scripts, is a label constructed from a word written in one script equivalent to a label constructed from the same word written in the other script? Is the answer the same for words in two different languages that translate into each other?

It is now generally understood that, in addition to the collision problems of possibly equivalent words and hence labels, it is possible to utilize characters that look alike -- "confusable" characters -- to spoof names in order to mislead or defraud users. That issue, driven by particular attacks such as those known as "phishing", has introduced stronger requirements for registry efforts to prevent problems than were previously generally recognized as important.

One commonly-proposed approach is to have a registry establish restrictions on the characters, and combinations of characters, it will permit to be included in a string to be registered as a label. Taking the Swedish top-level domain, .SE, as an example, a rule might be adopted that the registry "only accepts registrations in Swedish, using Latin script, and because of this, Unicode characters Latin-a, -b, -c,...". But, because there is not a 1:1 mapping between country and language, even a Country Code Top Level Domain (ccTLD) like .SE might have to accept registrations in other languages. For example, there may be a requirement for Finnish (the second most-used language in Sweden). What rules and code points are then defined for Finnish? Does it have special mappings that collide with those that are defined for Swedish? And what does one do in countries that use more than one script? (Finnish and Swedish use the same script.) In all cases, the dispute will ultimately be about whether two strings are the same (or confusingly similar) or not. That, in turn, will generate a discussion of how one defines "what is the same" and "what is similar enough to be a problem".

Another example arose recently that further illustrates the problem. If one were to use Cyrillic characters to represent the country code for Russia in a localized equivalent to the ccTLD label, the characters themselves would be indistinguishable from the Latin characters "P" and "Y" (in either lower- or uppercase) in most fonts. We presume this might cause some consternation in Paraguay.

These difficulties can never be completely eliminated by algorithmic means. Some of the problem can be addressed by appropriate tuning of the protocols and their tables, other parts by registry actions to reduce confusion and conflicts, and still other parts can be

addressed by careful design of user interfaces in application programs. But, ultimately, some responsibility to avoid being tricked or harmfully confused will rest with the user.

Another registry technique that has been extensively explored involves looking at confusable characters and confusion between complete labels, restricting the labels that can be registered based on relationships to what is registered already. Registries that adopt this approach might establish special mapping rules such as:

1. If you register something with code point A, domain names with B instead of A will be blocked from registration by others (where B is a character at a separate code point that has a confusingly similar appearance to A).

2. If you register something with code point A, you also get domain name with B instead of A.

These approaches are discussed in more detail for "CJK" characters in RFC 3743 [RFC3743] and more generally in RFC 4290 [RFC4290].

2.2.7. The IESG Statement and IDNA issues

The issues above, at least as they were understood at the time, provided the background for the IESG statement included in Section 1.6.1 (which, in turn, was part of the basis for the initial ICANN Guidelines) that a registry should have a policy about the scripts, languages, code points and text directions for which registrations will be accepted. While "accept all" might be an acceptable policy, it implies there is also a dispute resolution process that takes the problems listed above into account. This process must be designed for dealing with all types of potential disputes. For example, issues might arise between registrant and registry over a decision by the registry on collisions with already registered domain names and between registrant and trademark holder (that a domain name infringes on a trademark). In both cases, the parties disagreeing have different views on whether two strings are "equivalent" or not. They may believe that a string that is not allowed to be registered is actually different from one that is already registered. Or they might believe that two strings are the same, even though the rules adopted by the registry to prevent confusion define them as two different domain names.

3. Migrating to New Versions of Unicode

3.1. Versions of Unicode

While opinions differ about how important the issues are in practice, the use of Unicode and its supporting tables for IDNA appears to be far more sensitive to subtle changes than it is in typical Unicode applications. This may be, at least in part, because many other applications are internally sensitive only to the appearance of characters and not to their representation. Or those applications may be able to take effective advantage of script, language, or character class identification. The working group that developed IDNA concluded that attempting to encode any ancillary character information into the DNS label would be impractical and unwise, and the IAB, based in part on the comments in the ad hoc committee, saw no reason to review that decision.

The Unicode Consortium has sometimes used the likelihood of a combination of characters actually appearing in a natural language as a criterion for the safety of a possible change. However, as discussed above, DNS names are often fabrications -- abbreviations, strings deliberately formed to be unusual, members of a series sequenced by numbers or other characters, and so on. Consequently, a criterion that considers a change to be safe if it would not be visible in properly-constructed running text is not helpful for DNS purposes: a change that would be safe under that criterion could still be quite problematic for the DNS.

This sensitivity to changes has made it quite difficult to migrate IDNA from one version of Unicode to the next if any changes are made that are not strictly additive. A change in a code point assignment or definition may be extremely disruptive if a DNS label has been defined using the earlier form and any of its previous components has been moved from one table position or normalization rule to another. Unicode normalization tables, tables of scripts or languages and characters that belong to them, and even tables of confusable characters as an adjunct to security recommendations may be very helpful in designing registry restrictions on registrations and applications provisions for avoiding or identifying suspicious names. Ironically, they also extend the sensitivity of IDNA and its implementations to all forms of change between one version of Unicode and the next. Consequently, they make Unicode version migration more difficult.

An example of the type of change that appears to be just a small correction from one perspective but may be problematic from another was the correction to the normalization definition in 2004 [Unicode-PR29]. Community input suggested that the change would

cause problems for Stringprep, but the Unicode Technical Committee decided, on balance, that the change was worthwhile. Because of difficulties with consistency, some deployed implementations have decided to adopt the change and others have not, leading to subtle incompatibilities.

This situation leads to a dilemma. On the one hand, it is completely unacceptable to freeze IDNA at a Unicode version level that excludes more recently-defined characters and scripts that are important to those who use them. On the other hand, it is equally unacceptable to migrate from one version of Unicode to the next if such migration might invalidate an existing registered DNS name or some of its registered properties or might make the string or representation of that name ambiguous. If IDNA is to be modified to accommodate new versions of Unicode, the IETF will need to work with the Unicode Consortium and other bodies to find an appropriate balance in this area, but progress will be possible only if all relevant parties are able to fairly consider and discuss possible decisions that may be very difficult and unpalatable.

It would also prove useful if, during the course of that dialog, the need for Unicode Consortium concern with security issues in applications of the Unicode character set could be clarified. It would be unfortunate from almost every perspective considered here, if such matters slowed the inclusion of as yet unencoded scripts.

3.2. Version Changes and Normalization Issues

3.2.1. Unnormalized Combining Sequences

One of the advantages of the Unicode model of combining characters, as with previous systems that use character overstriking to accomplish similar purposes, is that it is possible to use sequences of code points to generate characters that are not explicitly provided for in the character set. However, unless sequences that are not explicitly provided for are prohibited by some mechanism (such as the normalization tables), such combining sequences can permit two related dangers.

- The first is another risk of character confusion, especially if the relationship of the combining character with characters it combines with are not precisely defined or unexpected combinations of combining characters are used. That issue is discussed in more detail, with an example, in Section 2.2.3.

- These same issues also inherently impact the stability of the normalization tables. Suppose that, somewhere in the world, there is a character that looks like a Roman-derived lowercase "i", but

with three (not one or two) dots above it. And suppose that the users of that character agree to represent it by combining a traditional "i" (U+0069) with a combining diaeresis (U+0308). So far, no problem. But, later, a broader need for this character is discovered and it is coded into Unicode either as a single precomposed character or, more likely under existing rules, by introducing a three-dot-above combining character. In either case, that version of Unicode should include a rule in NFKC that maps the "i"-plus-diaeresis sequence into the new, approved, one. If one does not do so, then there is arguably a normalization that should occur that does not. If one does so, then strings that were valid and normalized (although unanticipated) under the previous versions of Unicode become unnormalized under the new version. That, in turn, would impact IDNA comparisons because, effectively, it would introduce a change in the matching rules.

It would be useful to consider rules that would avoid or minimize these problems with the understanding that, for reasons given elsewhere, simply minimizing it may not be good enough for IDNA. One partial solution might be to ban any combination of a base character and a combining character that does not appear in a hypothetical "anticipated combinations" table from being used in a domain name label. The next subsection discusses a more radical, if impractical, view of the problem and its solutions.

3.2.2. Combining Characters and Character Components

For several reasons, including those discussed above, one thing that increases IDNA complexity and the need for normalization is that combining characters are permitted. Without them, complexity might be reduced enough to permit easier transitions to new versions. The community should consider the impact of entirely prohibiting combining characters from IDNs. While it is almost certainly unfeasible to introduce this change into Unicode as it is now defined and doing so would be extremely disruptive even if it were feasible, the thought experiment can be helpful in understanding both the issues and the implications of the paths not taken. For example, one consequence of this, of course, is that each new language or script, and several existing ones, would require that all of its characters have Unicode assignments to specific, precomposed, code points.

Note that this is not currently permitted within Unicode for Latin scripts. For non-Latin scripts, some such code points have been defined. The decisions that govern the assignment of such code points are managed entirely within the Unicode Consortium. Were the IETF to choose to reduce IDNA complexity by excluding combining characters, no doubt there would be additional input to the Unicode Consortium from users and proponents of scripts that precomposed

characters be required. The IAB and the IETF should examine whether it is appropriate to press the Unicode Consortium to revise these policies or otherwise to recommend actions that would reduce the need for normalization and the related complexities. However, we have been told that the Technical Committee does not believe it is reasonable or feasible to add all possible precomposed characters to Unicode. If Unicode cannot be modified to contain the precomposed characters necessary to support existing languages and scripts, much less new ones, this option for IDN restrictions will not be feasible.

3.2.3. When does normalization occur?

In many Unicode applications, the preferred solution is to pick a style of normalization and require that all text that is stored or transmitted be normalized to that form. (This is the approach taken in ongoing work in the IETF on a standard Unicode text form [net-utf8]). IDNA does not impose this requirement. Text is normalized and case-reduced at registration time, and only the normalized version is placed in the DNS. However, there is no requirement that applications show only the native (and lower-case where appropriate) characters associated with the normalized form in discussions or references such as URLs. If conventions used for all-ASCII DNS labels are to be extended to internationalized forms, such a requirement would be unreasonable, since it would prohibit the use of mixed-case references for clarity or market identification. It might even be culturally inappropriate. However, without that restriction, the comparison that will ultimately be made in the DNS will be between strings normalized at different times and under different versions of Unicode. The assertion that a string in normalized form under one version of Unicode will still be in normalized form under all future versions is not sufficient. Normalization at different times also requires that a given source string always normalizes to the same target string, regardless of the version under which it is normalized. That criterion is much more difficult to fulfill. The discussion above suggests that it may even be impossible.

Ignoring these issues with combining characters entirely, as IDNA effectively does today, may leave us "stuck" at Unicode 3.2, leading either to incompatibility differences in applications that otherwise use a modern version of Unicode (while IDN remains at Unicode 3.2) or to painful transitions to new versions. If decisions are made quickly, it may still be possible to make a one-time version upgrade to Version 4.1 or Version 5 of Unicode. However, unless we can impose sufficient global restrictions to permit smooth transitions, upgrading to versions beyond that one are likely to be painful (e.g., potentially requiring changing strings already in the DNS or even a new Punycode prefix) or impossible.

4. Framework for Next Steps in IDN Development

4.1. Issues within the Scope of the IETF

4.1.1. Review of IDNA

The IETF should consider reviewing RFCs 3454, 3490, 3491, and/or 3492, and update, replace, or supplement them to meet the criteria of this paragraph (one or more of them may prove impractical after further study). Any new versions or additional specifications should be adapted to the version of Unicode that is current when they are created. Ideally, they should specify a path for adapting to future versions of Unicode (some suggestions below may facilitate this). The IETF should also consider whether there are significant advantages to mapping some groups of characters, such as code points assigned to font variations, into others or whether clarity and comprehensibility for the user would be better served by simply prohibiting those characters. More generally, it appears that it would be worthwhile for the IETF to review whether the Unicode normalization rules now invoked by the Stringprep profile in Nameprep are optimal for the DNS or whether more restrictive rules, or an even more restrictive set of permitted character combinations, would provide better support for DNS internationalization.

The IAB has concluded that there is a consensus within the broader community that lists of code points should be specified by the use of an inclusion-based mechanism (i.e., identifying the characters that are permitted), rather than by excluding a small number of characters from the total Unicode set as Stringprep and Nameprep do today. That conclusion should be reviewed by the IETF community and action taken as appropriate.

We suggest that the individuals doing the review of the code points should work as a specialized design team. To the extent possible, that work should be done jointly by people with experience from the IETF and deep knowledge of the constraints of the DNS and application design, participants from the Unicode Consortium, and other people necessary to be able to reach a generally-accepted result. Because any work along these lines would be modifications and updates to standards-track documents, final review and approval of any proposals would necessarily follow normal IETF processes.

It is worth noting that sufficiently extreme changes to IDNA would require a new Punycode prefix, probably with long-term support for both the old prefix and the new one in both registration arrangements and applications. An alternative, which is almost certainly impractical, would be some sort of "flag day", i.e., a date on which the old rules are simultaneously abandoned by everyone and the new

ones adopted. However, preliminary analysis indicates that few, if any, of the changes recommended for consideration elsewhere in this document would require this type of version change. For example, suppose additional restrictions, such as those implied above, are imposed on what can be registered. Those restrictions might require policy decisions about how labels are to be disposed of if they conformed to the earlier rules but not to the new ones. But they would not inherently require changes in the protocol or prefix.

4.1.2. Non-DNS and Above-DNS Internationalization Approaches

The IETF should once again examine the extent to which it is appropriate to try to solve internationalization problems via the DNS and what place the many varieties of so-called "keyword systems" or other Internet navigational techniques might have. Those techniques can be designed to impose fewer constraints, or at least different constraints, than IDNA and the DNS. As discussed elsewhere in this document, IDNA cannot support information about scripts, languages, or Unicode versions on lookup. As a consequence of the nature of DNS lookups, characters and labels either match or do not match; a near-match is simply not a possible concept in the DNS. By contrast, observation of near-matching is common in human communication and in matching operations performed by people, especially when they have a particular script or language context in mind. The DNS is further constrained by a fairly rigid internal aliasing system (via CNAME and DNAME resource records), while some applications of international naming may require more flexibility. Finally, the rigid hierarchy of the DNS --and the tendency in practice for it to become flat at levels nearest the root-- and the need for names to be unique are more suitable for some purposes than others and may not be a good match for some purposes for which people wish to use IDNs. Each of these constraints can be relaxed or changed by one or more systems that would provide alternatives to direct use of the DNS by users. Some of the issues involved are discussed further in Section 5.3 and various ideas have been discussed in detail in the IETF or IRTF. Many of those ideas have even been described in Internet Drafts or other documents. As experience with IDNs and with expectations for them accumulates, it will probably become appropriate for the IETF or IRTF to revisit the underlying questions and possibilities.

4.1.3. Security Issues, Certificates, etc.

Some characters look like others, often as the result of common origins. The problem with these "confusable" characters, often incorrectly called homographs, has always existed when characters are presented to humans who interpret what is displayed and then make decisions based on what is seen. This is not a problem that exists only when working with internationalized domain names, but they make

the problem worse. The result of a survey that would explain what the problems are might be interesting. Many of these issues are mentioned in Unicode Technical Report #36 [UTR36].

In this and other issues associated with IDNs, precise use of terminology is important lest even more confusion result. The definition of the term 'homograph' that normally appears in dictionaries and linguistic texts states that homographs are different words that are spelled identically (for example, the adjective 'brief' meaning short, the noun 'brief' meaning a document, and the verb 'brief' meaning to inform). By definition, letters in two different alphabets are not the same, regardless of similarities in appearance. This means that sequences of letters from two different scripts that appear to be identical on a computer display cannot be homographs in the accepted sense, even if they are both words in the dictionary of some language. Assuming that there is a language written with Cyrillic script in which "cap" is a word, regardless of what it might mean, it is not a homograph of the Latin-script English word "cap".

When the security implications of visually confusable characters were brought to the forefront in 2005, the term homograph was used to designate any instance of graphic similarity, even when comparing individual characters. This usage is not only incorrect, but risks introducing even more confusion and hence should be avoided. The current preferred terminology is to describe these similar-looking characters as "confusable characters" or even "confusables".

Many people have suggested that confusable characters are a problem that must be addressed, at least in part, directly in the user interfaces of application software. While it should almost certainly be part of a complete solution, that approach creates it own set of difficulties. For example, a user switching between systems, or even between applications on the same system, may be surprised by different types of behavior and different levels of protection. In addition, it is unclear how a secure setup for the end user should be designed. Today, in the web browser, a padlock is a traditional way of describing some level of security for the end user. Is this binary signaling enough? Should there be any connection between a risk for a displayed string including confusable characters and the padlock or similar signaling to the user?

Many web browsers have adopted a convention, based on a "whitelist" or similar technique, of restricting the display of native characters to subdomains of top-level domains that are deemed to have safe practices for the registration of potentially confusable labels. IDNs in other domains are displayed as Punycode. These techniques may not be sufficiently sensitive to differences in policies among

top-level domains and their subdomains and so, while they are clearly helpful, they may not be adequate. Are other methods of dealing with confusable characters possible? Would other methods of identifying and listing policies about avoiding confusing registrations be feasible and helpful?

It would be interesting to see a more coordinated effort in establishing guidelines for user interfaces. If nothing else, the current whitelists are browser specific and both can, and do, differ between implementations.

4.1.4. Protocol Changes and Policy Implications

Some potential protocol or table changes raise important policy issues about what to do with existing, registered, names. Should such changes be needed, their impact must be carefully evaluated in the IETF, ICANN, and possibly other forums. In particular, protocol or policy changes that would not permit existing names to be registered under the newer rules should be considered carefully, balancing their importance against possible disruption and the issues of invalidating older names against the importance of consistency as seen by the user.

4.1.5. Non-US-ASCII in Local Part of Email Addresses

Work is going on in the IETF related to the local part of email addresses. It should be noted that the local part of email addresses has much different syntax and constraints than a domain name label, so to directly apply IDNA on the local part is not possible.

4.1.6. Use of the Unicode Character Set in the IETF

Unicode and the closely-related ISO 10646 are the only coded character sets that aspire to include all of the world’s characters. As such, they permit use of international characters without having to identify particular character coding standards or tables. The requirement for a single character set is particularly important for use with the DNS since there is no place to put character set identification. The decision to use Unicode as the base for IETF protocols going forward is discussed in [RFC2277]. The IAB does not see any reason to revisit the decision to use Unicode in IETF protocols.

4.2. Issues That Fall within the Purview of ICANN

4.2.1. Dispute Resolution

IDNs create new types of collisions between trademarks and domain names as well as collisions between domain names. These have impact on dispute resolution processes used by registries and otherwise. It is important that deployment of IDNs evolve in parallel with review and updating of ICANN or registry-specific dispute resolution processes.

4.2.2. Policy at Registries

The IAB recommends that registries use an inclusion-based model when choosing what characters to allow at the time of registration. This list of characters is in turn to be a subset of what is allowed according to the updated IDNA standard. The IAB further recommends that registries develop their inclusion-based models in parallel with dispute resolution process at the registry itself.

Most established policies for dealing with claimed or apparent confusion or conflicts of names are based on dispute resolution. Decisions about legitimate use or registration of one or more names are resolved at or after the time of registration on a case-by-case basis and using policies that are specific to the particular DNS zone or jurisdiction involved. These policies have generally not been extended below the level of the DNS that is directly controlled by the top-level registry.

Because of the number of conflicts that can be generated by the larger number of available and confusable characters in Unicode, we recommend that registration-restriction and dispute resolution policies be developed to constrain registration of IDNs and zone administrators at all levels of the DNS tree. Of course, many of these policies will be less formal than others and there is no requirement for complete global consistency, but the arguments for reduction of confusable characters and other issues in TLDs should apply to all zones below that specific TLD.

Consistency across all zones can obviously only be accomplished by changes to the protocols. Such changes should be considered by the IETF if particular restrictions are identified that are important and consistent enough to be applied globally.

Some potential protocol changes or changes to character-mapping tables might, if adopted, have profound registry policy implications. See Section 4.1.4.

4.2.3. IDNs at the Top Level of the DNS

The IAB has concluded that there is not one issue with IDNs at the top level of the DNS (IDN TLDs) but at least three very separate ones:

- If IDNs are to be entered in the root zone, decisions must first be made about how these TLDs are to be named and delegated. These decisions fall within the traditional IANA scope and are ICANN issues today.

- There has been discussion of permitting some or all existing TLDs to be referenced by multiple labels, with those labels presumably representing some understanding of the "name" of the TLD in different languages. If actual aliases of this type are desired for existing domains, the IETF may need to consider whether the use of DNAME records in the root is appropriate to meet that need, what constraints, if any, are needed, whether alternate approaches, such as those of [RFC4185], are appropriate or whether further alternatives should be investigated. But, to the extent to which aliases are considered desirable and feasible, decisions presumably must be made as to which, if any, root IDN labels should be associated with DNAME records and which ones should be handled by normal delegation records or other mechanisms. That decision is one of DNS root-level namespace policy and hence falls to ICANN although we would expect ICANN to pay careful attention to any technical, operational, or security recommendations that may be produced by other bodies.

- Finally, if IDN labels are to be placed in the root zone, there are issues associated with how they are to be encoded and deployed. This area may have implications for work that has been done, or should be done, in the IETF.

5. Specific Recommendations for Next Steps

Consistent with the framework described above, the IAB offers these recommendations as steps for further consideration in the identified groups.

5.1. Reduction of Permitted Character List

Generalize from the original "hostname" rules to non-ASCII characters, permitting as few characters as possible to do that job. This would involve a restrictive model for characters permitted in IDN labels, thus contrasting with the approach used to develop the original IDNA/Nameprep tables. That approach was to include all Unicode characters that there was not a clear reason to exclude.

The specific recommendation here is to specify such internationalized hostnames. Such an activity would fall to the IETF, although the task of developing the appropriate list of permitted characters will require effort both in the IETF and elsewhere. The effort should be as linguistically and culturally sensitive as possible, but smooth and effective operation of the DNS, including minimizing of complexity, should be primary goals. The following should be considered as possible mechanisms for achieving an appropriate minimum number of characters.

5.1.1. Elimination of All Non-Language Characters

Unicode characters that are not needed to write words or numbers in any of the world's languages should be eliminated from the list of characters that are appropriate in DNS labels. In addition to such characters as those used for box-drawing and sentence punctuation, this should exclude punctuation for word structure and other delimiters. While DNS labels may conveniently be used to express words in many circumstances, the goal is not to express words (or sentences or phrases), but to permit the creation of unambiguous labels with good mnemonic value.

5.1.2. Elimination of Word-Separation Punctuation

The inclusion of the hyphen in the original hostname rules is a historical artifact from an older, flat, namespace. The community should consider whether it is appropriate to treat it as a simple legacy property of ASCII names and not attempt to generalize it to other scripts. We might, for example, not permit claimed equivalents to the hyphen from other scripts to be used in IDNs. We might even consider banning use of the hyphen itself in non-ASCII strings or, less restrictively, strings that contained non-Latin characters.

5.2. Updating to New Versions of Unicode

As new scripts, to support new languages, continue to be added to Unicode, it is important that IDNA track updates. If it does not do so, but remains "stuck" at 3.2 or some single later version, it will not be possible to include labels in the DNS that are derived from words in languages that require characters that are available only in later versions. Making those upgrades is difficult, and will continue to be difficult, as long as new versions require, not just addition of characters, but changes to canonicalization conventions, normalization tables, or matching procedures (see Section 3.1). Anything that can be done to lower complexity and simplify forward transitions should be seriously considered.

5.3. Role and Uses of the DNS

We wish to remind the community that there are boundaries to the appropriate uses of the DNS. It was designed and implemented to serve some specific purposes. There are additional things that it does well, other things that it does badly, and still other things it cannot do at all. No amount of protocol work on IDNs will solve problems with alternate spellings, near-matches, searching for appropriate names, and so on. Registration restrictions and carefully-designed user interfaces can be used to reduce the risk and pain of attempts to do some of these things gone wrong, as well as reducing the risks of various sort of deliberate bad behavior, but, beyond a certain point, use of the DNS simply because it is available becomes a bad tradeoff. The tradeoff may be particularly unfortunate when the use of IDNs does not actually solve the proposed problem. For example, internationalization of DNS names does not eliminate the ASCII protocol identifiers and structure of URIs [RFC3986] and even IRIs [RFC3987]. Hence, DNS internationalization itself, at any or all levels of the DNS tree, is not a sufficient response to the desire of populations to use the Internet entirely in their own languages and the characters associated with those languages.

These issues are discussed at more length, and alternatives presented, in [RFC2825], [RFC3467], [INDNS], and [DNS-Choices].

5.4. Databases of Registered Names

In addition to their presence in the DNS, IDNs introduce issues in other contexts in which domain names are used. In particular, the design and content of databases that bind registered names to information about the registrant (commonly described as "whois" databases) will require review and updating. For example, the whois protocol itself [RFC3912] has no standard capability for handling non-ASCII text: one cannot search consistently for, or report, either a DNS name or contact information that is not in ASCII characters. This may provide some additional impetus for a switch to IRIS [RFC3981] [RFC3982] but also raises a number of other questions about what information, and in what languages and scripts, should be included or permitted in such databases.

6. Security Considerations

This document is simply a discussion of IDNs and IDNA issues; it raises no new security concerns. However, if some of its recommendations to reduce IDNA complexity, the number of available characters, and various approaches to constraining the use of confusable characters, are followed and prove successful, the risks of name spoofing and other problems may be reduced.

7. Acknowledgements

The contributions to this report from members of the IAB-IDN ad hoc committee are gratefully acknowledged. Of course, not all of the members of that group endorse every comment and suggestion of this report. In particular, this report does not claim to reflect the views of the Unicode Consortium as a whole or those of particular participants in the work of that Consortium.

The members of the ad hoc committee were: Rob Austein, Leslie Daigle, Tina Dam, Mark Davis, Patrik Faltstrom, Scott Hollenbeck, Cary Karp, John Klensin, Gervase Markham, David Meyer, Thomas Narten, Michael Suignard, Sam Weiler, Bert Wijnen, Kurt Zeilenga, and Lixia Zhang.

Thanks are due to Tina Dam and others associated with the ICANN IDN Working Group for contributions of considerable specific text, to Marcos Sanz and Paul Hoffman for careful late-stage reading and extensive comments, and to Pete Resnick for many contributions and comments, both in conjunction with his former IAB service and subsequently. Olaf M. Kolkman took over IAB leadership for this document after Patrik Faltstrom and Pete Resnick stepped down in March 2006.

Members of the IAB at the time of approval of this document were: Bernard Aboba, Loa Andersson, Brian Carpenter, Leslie Daigle, Patrik Faltstrom, Bob Hinden, Kurtis Lindqvist, David Meyer, Pekka Nikander, Eric Rescorla, Pete Resnick, Jonathan Rosenberg and Lixia Zhang.

8. References

8.1. Normative References

[ISO10646] International Organization for Standardization, "Information Technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane"", ISO/IEC 10646-1:2000, October 2000.

[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.

[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/).

8.2. Informative References

[DNS-Choices] Faltstrom, P., "Design Choices When Expanding DNS", Work in Progress, June 2005.

[ICANNv1] ICANN, "Guidelines for the Implementation of Internationalized Domain Names, Version 1.0", March 2003, <http://www.icann.org/general/idn-guidelines-20jun03.htm>.

[ICANNv2] ICANN, "Guidelines for the Implementation of Internationalized Domain Names, Version 2.0", November 2005, <http://www.icann.org/general/idn-guidelines-20sep05.htm>.

[IESG-IDN] Internet Engineering Steering Group (IESG), "IESG Statement on IDN", IESG Statements IDN Statement, February 2003, <http://www.ietf.org/IESG/STATEMENTS/IDNstatement.txt>.

[INDNS] National Research Council, "Signposts in Cyberspace: The Domain Name System and Internet Navigation", National Academy Press ISBN 0309-09640-5 (Book) 0309-54979-5 (PDF), 2005, <http://www7.nationalacademies.org/cstb/pub_dns.html>.

[ISO.2022.1986] International Organization for Standardization, "Information Processing: ISO 7-bit and 8-bit coded character sets: Code extension techniques", ISO Standard 2022, 1986.

[ISO.646.1991] International Organization for Standardization, "Information technology - ISO 7-bit coded character set for information interchange", ISO Standard 646, 1991.

[ISO.8859.2003] International Organization for Standardization, "Information processing - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1 (1998) - Part 2: Latin alphabet No. 2 (1999) - Part 3: Latin alphabet No. 3 (1999) - Part 4: Latin alphabet No. 4 (1998) - Part 5: Latin/Cyrillic alphabet (1999) - Part 6: Latin/ Arabic alphabet (1999) - Part 7: Latin/Greek alphabet (2003) - Part 8: Latin/Hebrew alphabet (1999) - Part 9: Latin alphabet No. 5 (1999) - Part 10: Latin alphabet No. 6 (1998) - Part 11: Latin/Thai alphabet (2001) - Part 13: Latin alphabet No. 7 (1998) - Part 14: Latin alphabet No. 8 (Celtic) (1998) - Part 15: Latin alphabet No. 9 (1999) - Part 16: Part 16: Latin alphabet No. 10 (2001)", ISO Standard 8859, 2003.

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

[RFC2825] IAB and L. Daigle, "A Tangled Web: Issues of I18N, Domain Names, and the Other Internet protocols", RFC 2825, May 2000.

[RFC3066] Alvestrand, H., "Tags for the Identification of Languages", BCP 47, RFC 3066, January 2001.

[RFC3467] Klensin, J., "Role of the Domain Name System (DNS)", RFC 3467, February 2003.

[RFC3536] Hoffman, P., "Terminology Used in Internationalization in the IETF", RFC 3536, May 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.

[RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, September 2004.

[RFC3981] Newton, A. and M. Sanz, "IRIS: The Internet Registry Information Service (IRIS) Core Protocol", RFC 3981, January 2005.

[RFC3982] Newton, A. and M. Sanz, "IRIS: A Domain Registry (dreg) Type for the Internet Registry Information Service (IRIS)", RFC 3982, January 2005.

[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.

[RFC4185] Klensin, J., "National and Local Characters for DNS Top Level Domain (TLD) Names", RFC 4185, October 2005.

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

[RFC4645] Ewell, D., "Initial Language Subtag Registry", RFC 4645, September 2006.

[RFC4646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 4646, September 2006.

[UTR] Unicode Consortium, "Unicode Technical Reports", <http://www.unicode.org/reports/>.

[UTR36] Davis, M. and M. Suignard, "Unicode Technical Report #36: Unicode Security Considerations", November 2005, <http://www.unicode.org/draft/reports/tr36/tr36.html>.

[UTR39] Davis, M. and M. Suignard, "Unicode Technical Standard #39 (proposed): Unicode Security Considerations", July 2005, <http://www.unicode.org/draft/reports/tr39/tr39.html>.

[Unicode-PR29] The Unicode Consortium, "Public Review Issue #29: Normalization Issue", Unicode PR 29, February 2004.

[Unicode10] The Unicode Consortium, "The Unicode Standard,

Version 1.0", 1991.

[W3C-Localization] Ishida, R. and S. Miller, "Localization vs. Internationalization", W3C International/ questions/qa-i18n.txt, December 2005.

[net-utf8] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", Work in Progress, April 2006.

Authors' Addresses

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

Phone: +1 617 491 5735
EMail: john-ietf@jck.com

Patrik Faltstrom
Cisco Systems

EMail: paf@cisco.com

Cary Karp
Swedish Museum of Natural History
Box 50007
Stockholm SE-10405
Sweden

Phone: +46 8 5195 4055
EMail: ck@nrm.museum

IAB

EMail: iab@iab.org

Full Copyright Statement

Intellectual Property

Acknowledgement

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