Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 16, 2009

J. Klensin

September 12, 2008

Internationalized Domain Names for Applications (IDNA): Definitions, Background and Rationale
draft-ietf-idnabis-rationale-02.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on March 16, 2009.

Abstract

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

Table of Contents

1. Introduction

1.1. Context and Overview

Several years have passed since the original protocol for Internationalized Domain Names (IDNs) was completed and deployed. During that time, a number of issues have arisen, including a subset of those described in a recent IAB report [RFC4690] and the need to update the system to deal with newer versions of Unicode. 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). Some tuning of the existing protocols and the tables on which they depend is now required. Where it is important to understanding of the revised protocols, this document further explains the issues that have been encountered. It also provides an overview of the new IDNA model and explanatory material for it. Additional explanatory material for the specific components of the proposals will appear with the associated documents.

1.2. Discussion Forum

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

This work is being discussed in the IETF "idnabis" Working Group and 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" 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 is possible even under the original rules where only ASCII characters were permitted. However, one can hope, through the application of different techniques at different points (see Section 5.3), 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 within the context of applications in the domain name space.

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 front-end processing and more specifically in the user interface (see Section 8).

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

after access to the DNS, 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 can be used as a last

resort to allow rudimentary IDN usage. For example, they might be the best choice for display if it were known that relevant fonts were not available on the user's computer. In order to allow user-friendly input and output of the IDNs and acceptance of some characters as equivalent to those to be processed according to the protocol, the applications need to be modified to conform to this specification.

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

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 "IDNA2008" is used to refer to a new version of IDNA as described in this document and in the documents described in Section 4. References to "these specifications" are to the entire set.

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 [Unicode51] is a coded character set containing almost 100,000 characters as of the current version. 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 four to six digit 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 0000..007F. Unicode may be thought of as an extension of ASCII; it includes all the ASCII characters and associates them with equivalent code points.

"Letters" are, informally, generalizations from the ASCII and common-sense understanding of that term, i.e., characters that are used to write text that are not digits, symbols, or punctuation. Formally, they are characters with a Unicode General Category value starting in

"L" (see Section 4.5 of [Unicode51]).

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" is used to describe the combination of operations performed by this protocol and those actually performed by a DNS

resolver. The process of placing an entry into the DNS is referred to as "registration", similar to common contemporary usage in other contexts. Consequently, any DNS zone administration is described as a "registry", regardless of the actual administrative arrangements or level in the DNS tree. A note about that relationship is included in the text below where it seems particularly significant.

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, 0030..0039, 0041..005A, and 0061..007A. "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. Lack of clarity about that terminology has contributed to confusion about intent in some cases. This document generally uses the term "domain name". When it refers to, e.g., host name syntax restrictions, it explicitly cites the relevant defining documents. The remaining definitions in this subsection are essentially a review.

A label is an individual component 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 RFC 1123 [RFC1123], which can be explicit as in "www.example.com." or implicit as in "www.example.com", is not considered in this specification.) IDNA extends the set of usable characters in labels that are treated as text (as distinct from the binary string labels discussed in RFC 1035 and RFC 2181 [RFC2181] and the bitstring ones described in RFC 2673 [RFC2673]). For the rest of this document and in the related ones, the term "label" is shorthand for "text label", and "every label" means "every text label".

1.5.4. Terminology Specific to IDNA

This section defines some terminology to reduce dependence on terms and definitions that have been problematic in the past.

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 in this subsection. In the next, it defines a historical one to be slightly more precise for IDNA contexts.

- A string is "IDNA-valid" if it meets all of the requirements of these specifications for an IDNA label. IDNA-valid strings may appear in either of two forms, defined immediately below. 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, see Section 1.5.4.5) form of an IDNA-valid string. It must be a complete label: IDNA is defined for labels, not for parts of them and not for complete domain names. 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 rules for the preferred form described in RFC 1034, RFC 1035, and RFC 1123.

- A "U-label" is an IDNA-valid string of Unicode characters, including at least one non-ASCII character, expressed in a standard Unicode Encoding Form -- normally UTF-8 in an Internet transmission context -- and subject to the constraint below. Conversions between U-labels and A-labels are performed according to the "Punycode" specification [RFC3492], adding or removing the ACE prefix (see Section 1.5.4.5) as needed.

To be valid, U-labels and A-labels must obey an important symmetry constraint. While that constraint may be tested in any of several ways, an A-label must be capable of being produced by conversion from a U-label and a U-label must be capable of being produced by conversion from an A-label. Among other things, this implies that both U-labels and A-labels must be strings in Unicode NFC [Unicode-UAX15] normalized form. These strings MUST contain only characters specified elsewhere in this document and its companion documents, and only in the contexts indicated as appropriate.

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. For the U-label, constraints imposed by existing protocols and their presentation forms make the

length restriction apply to the length in octets of the UTF-8 form of those labels (which will always be greater than or equal to the length in code points). 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 (see the next subsection) 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 the other two, in presentation and user interface forms and in selected protocols other than those of the DNS itself. Strings that do not conform to the rules for one of these three categories and, in particular, strings that contain "--" in the third and fourth character position but are:

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

are invalid in IDNA-conformant applications as labels in domain names that identify Internet hosts or similar resources. This restriction on strings containing "--" is required for three reasons:

- to prevent confusion with pre-IDNA coding forms;
- to permit future extensions that would require changing the prefix, no matter how unlikely those might be (see Section 9.2.3); and
- to reduce the opportunities for attacks via the encoding system.

1.5.4.2. LDH-label and Internationalized Label

In the hope of further clarifying discussions about IDNs, these specifications use the term "LDH-label" strictly to refer to an all-ASCII label that obeys the preferred syntax (often known as "hostname" (from RFC 952 [RFC0952]) or "LDH") conventions and that is not an IDN. It should be stressed that an A-label obeys the "hostname" rules and is sometimes described as "LDH-conformant" or in similar language but that it is not an LDH-label as used in this document.

1.5.4.3. Internationalized Domain Name

An "internationalized domain name" (IDN) is a domain name that may contain any mixture of LDH-labels, A-labels, or U-labels. This implies that every conventional domain name is an IDN (which implies that it is possible for a domain name to be an IDN without it containing any non-ASCII characters). 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. Because of the diversity of characters that can be used in a U-label and the confusion they might cause, such restrictions are mandatory for IDN registries and zones even though the particular restrictions are not part of these specifications. Because these restrictions, commonly known as "registry restrictions", only affect what can be registered and not lookup processing, they 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. See Section 5.2.

"Internationalized label" is used when a term is needed to refer to a single label of an IDN, i.e., one that might be any of an LDH-label, A-label, or U-label. 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.

1.5.4.4. 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.

1.5.4.5. 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".

1.5.4.6. 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() or getaddrinfo() standard C library functions; the part of an email address following the at-sign (@) in the parameter to the SMTP MAIL or RCPT commands or 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 ACE prefix and about whether it is required that such strings could have been the output of the ToASCII operation

(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 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.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.6. Comprehensibility of IDNA Mechanisms and Processing

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).

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 enable 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 and reconciliation of differences in character forms 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 and, because display functions are rarely controlled by the types of applications that would call upon IDNA, will rarely be very effective.

However, shifting responsibility for character mapping and other adjustments from the protocol (where it was located in IDNA2003) to the user interface or processing before invoking IDNA raises issues about both what that processing should do and about compatibility for references prepared in an IDNA2003 context. Those issues are discussed in Section 8.

Operations for converting between local character sets and normalized Unicode are part of this general set of user interface issues. The conversion is obviously not required at all in a Unicode-native system that maintains all strings in Normalization Form C (NFC). It may, however, involve some complexity in a system that is not Unicode-native, especially if the elements of the local character set do not map exactly and unambiguously into Unicode characters or do so in a way that is not 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. If a local operating system uses one of the ISO 8859 character sets or an extensive national or industrial system such as GB18030 [GB18030] or BIG5 [BIG5], one must correctly identify the character set in use before converting to Unicode even though those character coding systems are substantially or completely Unicode-compatible (i.e., all of the code points in them have an exact and unique mapping to Unicode code points). It may be even more difficult when the character coding system in local use is based on conceptually different assumptions than those used by Unicode about, e.g., about font encodings used for publications in some Indic scripts. Those differences may not easily yield unambiguous conversions or interpretations even if each coding system is internally consistent and adequate to represent the local language and script.

2. The Revised 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.

[[anchor17: This paragraph is somewhat redundant with material above.It will be dropped in -03 if there are not strong arguments for keeping it here.]]

3. Processing in IDNA2008

These specifications separate Domain Name Registration and Lookup in the protocol specification. Doing so reflects current practice in

which per-registry restrictions and special processing are applied at registration time but not during lookup. Even more important in the longer term, it facilitates incremental addition of permitted character groups to avoid freezing on one particular version of Unicode.

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

4. IDNA2008 Document List

[[anchor19: This section will need to be extensively revised or removed before publication.]]

The following documents are being produced as part of the IDNA2008 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 [IDNA2008-Protocol].
- A document describing the "Bidi problem" with Stringprep and proposing a solution [IDNA2008-Bidi].
- A specification of the categories and rules that identify the code points allowed in a U-label, based on Unicode 5.0 code assignments. See Section 5 and [IDNA2008-Tables].
- 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 be IETF products and may be prepared and completed asynchronously with those described above.

5. Permitted Characters: An Inclusion List

This section provides an overview of the model used to establish the algorithm and character lists of [IDNA2008-Tables] and describes the names and applicability of the categories used there. Note that the inclusion of a character in the first category group does not imply that it can be used indiscriminately; some characters are associated with contextual rules that must be applied as well.

The information given in this section is provided to make the rules, tables, and protocol easier to understand. It is not normative. The normative generating rules appear in [IDNA2008-Tables] and the rules that actually determine what labels can be registered or looked up are in [IDNA2008-Protocol].

5.1. A Tiered Model of Permitted Characters and Labels

Moving to an inclusion model requires respecifying the list of characters that are permitted in IDNs. In IDNA2003, the role and utility of characters are independent of context and fixed forever (or until the standard is replaced). Making completely context-independent rules globally has proven impractical because some characters, especially those that are called "Join_Controls" in Unicode, are needed to make reasonable use of some scripts but have no visible effect(s) in others. Of necessity, IDNA2003 prohibited those types of characters entirely. But the restrictions were much too severe to permit an adequate range of mnemonics for terminology based on some languages. The requirement to support those characters but limit their use to very specific contexts was reinforced by the observation that handling of particular characters across the languages that use a script, or the use of similar or identical-looking characters in different scripts, is less well understood than many people believed it was several years ago.

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

5.1.1. PROTOCOL-VALID

Characters identified as "PROTOCOL-VALID" (often abbreviated "PVALID") are, in general, permitted by IDNA for all uses in IDNs. Their use may be restricted by rules about the context in which they

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

The term "PROTOCOL-VALID" is used to stress the fact that the presence of a character in this category does not imply that a given registry need accept registrations containing any of the characters in the category. Registries are still expected to apply judgment about labels they will accept and to maintain rules consistent with those judgments (see [IDNA2008-Protocol] and Section 5.3).

Characters that are placed in the "PROTOCOL-VALID" category are never removed from it unless the code points themselves are removed from Unicode (such removal would be inconsistent with the Unicode stability principles (see [Unicode51], Appendix F) and hence should never occur).

[[anchor21: Placeholder: Does this topic or comment need additional discussion or explanation?]]

5.1.1.1. Contextual Rules

Some characters may be unsuitable for general use in IDNs but necessary for the plausible support of some scripts. The two most commonly-cited examples are the zero-width joiner and non-joiner characters (ZWJ, U+200D and ZWNJ, U+200C), but provisions for unambiguous labels may require that other characters be restricted to particular contexts. For example, the ASCII hyphen is not permitted to start or end a label, whether that label contains non-ASCII characters or not.

These characters must not appear in IDNs without additional restrictions, typically because they have no visible consequences 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, they are specially identified as "CONTEXTUAL RULE REQUIRED" and, when adequately understood, associated with a rule. In addition, the rule will define whether it is to be applied on lookup as well as registration. A distinction is made between characters that indicate or prohibit joining (known as "CONTEXT-JOINER" or "CONTEXTJ") and other characters requiring contextual treatment ("CONTEXT-OTHER" or "CONTEXTO"). Only the former are fully tested at lookup time.

5.1.1.2. Rules and Their Application

The actual rules may be present or absent. If present, they may have values of "True" (character may be used in any position in any label), "False" (character may not be used in any label), or may be a set of procedural rules that specify the context in which the character is permitted.

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

Because it is easier to identify these characters than to know that they are actually needed in IDNs or how to establish exactly the right rules for each one, a rule may have a null value in a given version of the tables. Characters associated with null rules 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.

The description of the syntax of the rules, and the rules themselves, appears in [IDNA2008-Tables].

5.1.2. DISALLOWED

Some characters are sufficiently problematic for use in IDNs that they should be excluded for both registration and lookup (i.e., IDNA-conforming applications performing name lookup should verify that these characters are absent; if they are present, the label strings should be rejected rather than converted to A-labels and looked up.

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

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

and replace the relevant RFC with one containing appropriate exceptions.

[[anchor23: Note in Draft: the permanence of DISALLOWED was still under discussion in the WG when this draft was posted. The text above reflects the editor's opinion about the emerging consensus but is subject to change as the discussion continues.]]

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

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

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

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

5.1.3. UNASSIGNED

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

5.2. Registration Policy

While these recommendations cannot and should not define registry policies, registries SHOULD develop and apply additional restrictions to reduce confusion and other problems. For example, it is generally believed that labels containing characters from more than one script

are a bad practice although there may be some important exceptions to that principle. Some registries may choose to restrict registrations to characters drawn from a very small number of scripts. For many scripts, the use of variant techniques such as those as described in [RFC3743] and [RFC4290], and illustrated for Chinese by the tables described in RFC 4713 [RFC4713] may be helpful in reducing problems that might be perceived by users. It is worth stressing that these principles of policy development and application apply at all levels of the DNS, not only, e.g., TLD registrations and that even a trivial, "anything permitted that is valid under the protocol" policy is helpful in that it helps users and application developers know what to expect..

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

The essence of the character rules in IDNA2008 is based on the realization that there is no magic bullet for any of the issues associated with a multiscript DNS. Instead, the specifications define a variety of approaches that, together, constitute multiple lines of defense against ambiguity in identifiers and loss of referential integrity. 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 by a protocol alone. As discussed in the previous section (Section 5.2), registries are expected to restrict what they permit to be registered, devising and using rules that are designed to optimize the balance between confusion and risk on the one hand and maximum expressiveness in mnemonics on the other.

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

6. Issues that Constrain Possible Solutions

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 that contains characters that are normally written right to left is discussed in [IDNA2008-Bidi]. In particular, there are 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, including the current one, must be clear about the network (transmission on the wire) order of characters in labels and for the labels in complete (fully-qualified) domain names. In order to prevent user confusion and, in particular, to reduce the chances for inconsistent transcription of domain names from printed form, 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 or specified by the operating system, 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 of A-labels, but are encouraged to not do so except as an interface for special purposes, possibly for debugging, or to cope with display limitations. In general, they SHOULD allow, but not encourage, user input of that label form. A-labels are opaque and ugly, and, where possible, should thus only be exposed to users and in contexts in which they are absolutely needed. Because IDN labels can be rendered either as A-labels or U-labels, the application may reasonably have an option for the user to select the preferred method of display; 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. This provision is intended to prevent situations in which, e.g., UTF-8 domain names appear embedded in text that is otherwise in some other character coding.

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

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

Users often have expectations about character matching or equivalence that are based on their languages and the orthography of those

languages. These expectations may not be consistent with forms or actions that can be naturally accommodated in a character coding system, especially if multiple languages are written using the same script but using different conventions. A Norwegian user might expect a label with the ae-ligature to be treated as the same label

as one using the Swedish spelling with a-umlaut even though applying that mapping to English would be astonishing to users. A user in German might expect a label with an o-umlaut and a label that had "oe" substituted, but was otherwise the same, treated as equivalent even though that substitution would be a clear error in Swedish. A Chinese user might expect automatic matching of Simplified and Traditional Chinese characters, but applying that matching for Korean or Japanese text would create considerable confusion. For that matter, an English user might expect "theater" and "theatre" to match.

Related issues arise because 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). When these ligature and digraph forms have the same interpretation across all languages that use a given script, application of Unicode normalization generally resolves the differences and causes them to match. When they have different interpretations, any requirements for matching must utilize other methods or users must be educated to understand that matching will not occur.

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 for the "umlauted a" character. The inverse is however not true, and those two characters cannot necessarily be combined into an "umlauted a". This also applies to another German character, the "umlauted o" (U+00F6 LATIN SMALL LETTER O WITH DIAERESIS) which, for example, cannot be used for writing the name of the author "Goethe". It is also a letter in the Swedish alphabet where, like the "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.

Some of the ligatures that have explicit code points in Unicode were given special handling in IDNA2003 and now pose additional problems as people argue that they should have been treated differently to preserve important information. For example, the German character Eszett (Sharp S, U+00DF) is retained as itself by NFKC but case-folded by Stringprep to "ss", but the closely-related, but less frequently seen, character "Long S T" (U+FB05) is a compatibility character that is mapped out by NFKC. Unless exceptions are made, both will be treated as DISALLOWED by IDNA2008. But there is significant interest in an exception, especially for Eszett. Depending on what the exception was, making it would either raise some backward compatibility problems with IDNA2003 or create an unusual special case that would highlight differences in preferred orthography between German as written in Germany and German as written in some other countries, notably Switzerland. Additional discussion of issues with Eszett appear in Section 9.2.7.

Additional cases with alphabets written right to left are described in Section 6.5.

Whether ligatures and digraphs are to be treated as a sequence of characters or as a single standalone one constitute a problem that cannot be resolved solely by operating on scripts. They are, however, a key concern in the IDN context. Their 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. Case Mapping and Related Issues

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

The solution of keeping the characters separate but doing matching independent of case is not feasible with an IDNA-like model because the matching would then have to be done on the server rather than have characters mapped on the client. That situation was recognized in IDNA2003 and nothing in IDNA2008 fundamentally changes it or could do so. In IDNA2003, all upper-case characters are mapped to lower-case ones and, in general, all code points that represent alternate forms of the same character are mapped to that character (including mapping Greek final form sigma to the medial form). IDNA2008 permits, at the risk of some incompatibility, slightly more flexibility in this area. That additional flexibility still does not solve the problem with final form sigma and other characters that Unicode treats as completely separate characters that match only under casemapping if at all. Many people now believe these should be handled as separate characters so information about them can be preserved in the transformations to A-labels and back. However making a change to permit that behavior would create a situation in which the same string, valid in both protocols, would be interpreted differently by IDNA2003 and IDNA2008. In principle, that would violate one of the conditions discussed in Section 9.2.3.1 and hence

require a prefix change. Of course, if a prefix change were made (at the costs discussed in Section 9.2.3.3) there would be several options, including, if desired, assigning the characer to the CONTEXTUAL RULE REQUIRED category and requiring that it only be used in carefully-selected contexts.

6.5. Right to Left Text

In order to be sure that the directionality of right to left text is unambiguous, IDNA2003 required that any label in which right to left characters appear both starts and ends with them, 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. The algorithmic model used in IDNA2003 rejects the label when the final character in a right to left string requires a combining mark in order to be correctly represented.

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). The new rules for right to left scripts are described in [IDNA2008-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, not only must the protocol be carefully designed and implemented, but zone administrators (registries) must have and require sensible policies about what is

registered -- conservative policies -- and implement and enforce them.

Conversely, lookup applications 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 elsewhere, if a string cannot be successfully found in the DNS after the lookup processing described here, it makes no difference whether it simply wasn't registered or was prohibited by some rule.

If lookup applications, as a user interface (UI) or other local 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 reinforce the good choices and discourage the poor ones. In this context, a lookup application that decides a string that is valid under the protocol is dangerous and refuses to look it up is in violation of the protocols; one that is willing to look something up, but warns against it, is exercising a local choice.

8. Front-end and User Interface Processing

Domain names may be identified and processed in many contexts. They may be typed in by users either by themselves or as part of URIs or IRIs. They may occur in running text or be processed by one system after being provided in another. Systems may wish to try to normalize URLs so as to determine (or guess) whether a reference is valid or two references point to the same object without actually looking the objects up and comparing them (that is necessary, not just a choice, for URI types that are not intended to be resolved). Some of these goals may be more easily and reliably satisfied than others. While there are strong arguments for any domain name that is

placed "on the wire" -- transmitted between systems -- to be in the minimum-ambiguity forms of A-labels, U-labels, or LDH-labels, it is inevitable that programs that process domain names will encounter variant forms.

One source of such forms will be labels created under IDNA2003 because that protocol allowed labels that were transformed before they were turned from native-character into ACE ("xn--...") format. One consequence of the transformations was that, when the ToUnicode and ToASCII operations of IDNA2003 were applied, ToUnicode(ToASCII(original-label)) often did not produce the original-label. IDNA2008 explicitly defines A-labels and U-labels as different forms of the same abstract label, forms that are stable when conversions are performed between them, without mappings. A different way of explaining this is that there are, today, domain names in files on the Internet that use characters that cannot be represented directly in, or recovered from, (A-label) domain names but for which interpretations are provided by IDNA2003. There are two major categories of such characters, those that are removed by NFKC normalization and those upper-case characters that are mapped to

lower-case (there are also a few characters that are given special-case mapping treatment in Stringprep).

Other issues in domain name identification and processing arise because IDNA2003 specified that several other characters be treated as equivalent to the ASCII period (dot, full stop) character used as a label separator. If a string that might be a domain name appears in an arbitrary context (such as running text), it is difficult, even with only ASCII characters, to know whether an actual domain name (or a protocol parameter like a URI) is present and where it starts and ends. When using Unicode, this gets even more difficult if treatment of certain special characters (like the dot that separates labels in a domain name) depends on context (e.g., prior knowledge of whether the string represents a domain name or not). That knowledge is not available if the primary heuristic for identifying the presence of domain names in strings depends on the presence of dots separating groups of characters with no intervening spaces.
[[anchor27: Above text is a substitute for an earlier (pre -01) version and is hoped to be more clear. Comments and improvements welcome.]]

As discussed elsewhere in this document, the IDNA2008 model removes all of these mappings and interpretations, including the equivalence of different forms of dots, from the protocol, discouraging such mappings and leaving them, when necessary, to local processing. This should not be taken to imply that local processing is optional or can be avoided entirely. Instead, unless the program context is such that it is known that any IDNs that appear will be either U-labels or A-labels, or that other forms can safely be rejected, some local processing of apparent domain name strings will be required, both to maintain compatibility with IDNA2003 and to prevent user astonishment. Such local processing, while not specified in this document or the associated ones, will generally take one of two forms:

- Generic Preprocessing.
  When the context in which the program or system that processes domain names operates is global, a reasonable balance must be found that is sensitive to the broad range of local needs and assumptions while, at the same time, not sacrificing the needs of one language, script, or user population to those of another.

  For this case, the best practice will usually be to apply NFKC and case-mapping (or, perhaps better yet, Stringprep itself), plus dot-mapping where appropriate, to the domain name string prior to applying IDNA. That practice will not only yield a reasonable compromise of user experience with protocol requirements but will be almost completely compatible with the various forms permitted

  by IDNA2003.

- Highly Localized Preprocessing.
  Unlike the case above, there will be some situations in which software will be highly localized for a particular environment and carefully adapted to the expectations of users in that environment. The many discussions about using the Internet to preserve and support local cultures suggest that these cases may be more common in the future than they have been so far.

  In these cases, we should avoid trying to tell implementers what they should do, if only because they are quite likely (and for good reason) to ignore us. We would assume that they would map characters that the intuitions of their users would suggest be

mapped and would hope that they would do that mapping as early as possible, storing A-label or U-label forms in files and transporting only those forms between systems. One can imagine switches about whether some sorts of mappings occur, warnings before applying them or, in a slightly more extreme version of the approach taken in Internet Explorer version 7 (IE7), systems that utterly refuse to handle "strange" characters at all if they appear in U-label form. None of those local decisions are a threat to interoperability as long as (i) only U-labels and A-labels are used in interchange with systems outside the local environment, (ii) no character that would be valid in a U-label as itself is mapped to something else, (iii) any local mappings are applied as a preprocessing step (or, for conversions from U-labels or A-labels to presentation forms, postprocessing), not as part of IDNA processing proper, and (iv) appropriate consideration is given to labels that might have entered the environment in conformance to IDNA2003. [[anchor28: Placeholder: there have been suggestions that this text be removed entirely. Comments (or improved text) welcome.]]

In either case, it is vital that user interface designs and, where the interfaces are not sufficient, users, be aware that the only forms of domain names that this protocol anticipates will resolve globally or compare equal when crude methods (i.e., those not conforming to Section 1.5.4.4) are used are those in which all native-script labels are in U-label form. Forms that assume mapping will occur, especially forms that were not valid under IDNA2003, may or may not function in predictable ways across all implementations.

9. Relationship to IDNA2003 and Earlier Versions of Unicode

9.1. Summary of Major Changes from IDNA2003

1. Update base character set from Unicode 3.2 to Unicode version-agnostic.

2. Separate the definitions for the "registration" and "lookup" activities.

3. Disallow symbol and punctuation characters except where special exceptions are necessary.

4. Remove the mapping and normalization steps from the protocol and have them instead done by the applications themselves, possibly in a local fashion, before invoking the protocol.

5. Change the way that the protocol specifies which characters are allowed in labels from "humans decide what the table of codepoints contains" to "decision about codepoints are based on Unicode properties plus a small exclusion list created by humans".

6. Introduce the new concept of characters that can be used only in specific contexts.

7. Allow typical words and names in languages such as Dhivehi and Yiddish to be expressed.

8. Make bidirectional domain names (delimited strings of labels, not just labels standing on their own) display in a non-surprising fashion whether they appear in obvious domain name contexts or as part of running text in paragraphs.

9. Remove the dot separator from the mandatory part of the protocol.

10. Make some currently-valid labels that are not actually IDNA labels invalid.

9.2. Migration and Version Synchronization

9.2.1. 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 or, subsequent to this version, "heavy" processes such as formation of an IETF WG. 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.

9.2.1.1. General IDNA Validity Criteria

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", marks needed to form letters, numerals, or other code points used to write words in some language. Symbols, drawing characters, and various notational characters are 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 inclusion and the complexities that would come with it (additional discussion and rationale for the symbol decision appears in Section 9.2.5).

- 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 certainly not prohibited if the conventions and orthography of a particular language cause that to be possible). Even for English, very common constructions -- contractions like "don't" or "it's", names that are written with apostrophes such as "O'Reilly" or characters for which apostrophes are common substitutes, and words whose usually-preferred spellings retain diacritical marks from earlier forms -- cannot be represented in DNS labels.

- Characters that are unassigned (have no character assignment at all) in the version of Unicode being used by the registry or application are not permitted, even on lookup. There are at least two reasons for this. Tests involving the context of characters (e.g., some characters being permitted only adjacent to ones of specific types but otherwise invisible or very problematic for other reasons) and integrity tests on complete labels are needed. Unassigned code points cannot be permitted because one cannot determine whether particular code points will require contextual rules (and what those rules should be) before characters are assigned to them and the properties of those characters fully understood. Second, Unicode specifies that an unassigned code point normalizes and case folds to itself. If the code point is later assigned to a character, and particularly if the newly-

assigned code point has a combining class that determines its placement relative to other combining characters, it could normalize to some other code point or sequence, creating confusion and/or violating other rules listed here.

- 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 lookup). 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 (described in more precise form in [IDNA2008-Tables]). The principles are not just an interpretation of the tables.

9.2.1.2. Labels in Registration

Anyone entering a label into a DNS zone must properly validate that label -- i.e., be sure that the criteria for that label are met -- in order for applications to work as intended. This principle is not new: for example, zone administrators are expected to verify that names meet "hostname" [RFC0952] or special service location formats [RFC2782] where necessary for the expected applications. For zones that will contain IDNs, support for Unicode version-independence requires restrictions on all strings placed in the zone. In particular, for such zones:

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

- The Unicode tables (i.e., tables of code points, character classes, and properties) and IDNA tables (i.e., tables of contextual rules such as those 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.

[[anchor31: Note in draft: the above text was changed significantly between -00 and -01 to clearly restrict its scope to zones supporting

IDNA and to eliminate comments about labels containing "--" in the third and forth positions but with different prefixes. There appears to be consensus that more extensive rules belong in a "best practices" document about appropriate DNS labels, but that document is not in-scope for the IDNABIS WG.]]

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.

Systems looking up or resolving DNS labels, especially IDN DNS labels, MUST be able to assume that applicable registration rules were followed for names entered into the DNS.

9.2.1.3. Labels in Lookup

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.

- MUST validate the characters in 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 "DISALLOWED") or code points that are unassigned in its version of Unicode.

- MUST validate the label itself for conformance with a small number of whole-label rules, notably verifying that there are no leading combining marks, that the "bidi" conditions are met if right to left characters appear, that any required contextual rules are available and that, if such rules are associated with Joiner Controls, they are tested.

- o MUST NOT validate other 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 lookup application becomes less sensitive to version incompatibilities with the particular zone registry associated with the domain name.

An application or client that looks processes names according to this protocol and then resolves them in the DNS will be able to locate any

name that is validly 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).

9.2.2. More Flexibility in User Agents

These specifications do not perform mappings between one character or code point and others for any reason. Instead, they prohibits the characters that would be mapped to others by normalization, case folding, or other rules. As examples, while mathematical characters based on Latin ones are accepted as input to IDNA2003, they are prohibited in IDNA2008. Similarly, double-width characters and other variations are prohibited as IDNA input.

Since the rules in [IDNA2008-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 as input to the lookup process. It is perfectly reasonable for systems that support user interfaces to perform some

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

One specific, and very important, instance of this strategy arises with case-folding. In the ASCII-only DNS, names are looked up and matched in a case-independent way, but no actual case-folding occurs. Names can be placed in the DNS in either upper or lower case form (or any mixture of them) and that form is preserved, returned in queries, and so on. IDNA2003 simulated 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 although the specifications require that it be performed prior to invoking the protocol) 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., be present in a U-label), then IDNA2008 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) expecting 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 them when they are not natural to the localized environment.

9.2.3. 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 and the implications of such a change.

9.2.3.1. Conditions Requiring a Prefix Change

An IDN prefix change is needed if a given string would be looked up or otherwise interpreted differently depending on the version of the protocol or tables being used. 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 IDNA2008.

2. An input string that is valid under IDNA2003 and also valid under IDNA2008 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 9.2.3.2, 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.

9.2.3.2. 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 affect characters that were already invalid under IDNA2003.

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

Of course, because these specifications do not involve changes to Stringprep or Nameprep, the third condition above and part of the second are moot.

9.2.3.3. Implications of Prefix Changes

While it might be possible to make a prefix change, the costs of such a change are considerable. Even if they wanted to do so, all registries could not convert all IDNA2003 ("xn--") registrations to a new form at the same time and synchronize that change with applications supporting lookup. Unless all existing registrations were simply to be declared invalid, and perhaps even then, systems that needed to support both labels with old prefixes and labels with new ones would first process a putative label under the IDNA2008 rules and try to look it up and then, if it were not found, would

process the label under IDNA2003 rules and look it up again. That process could significantly slow down all processing that involved IDNs in the DNS especially since, in principle, a fully-qualified name could contain a mixture of labels that were registered with the old and new prefixes, a situation that would make the use of DNS caching very difficult. In addition, looking up the same input string as two separate A-labels would create some potential for confusion and attacks, since they could, in principle, map to

different targets and then resolve to different DNS label nodes.

Consequently, a prefix change is to be avoided if at all possible, even if it means accepting some IDNA2003 decisions about character distinctions as irreversible.

9.2.4. 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 9.2.3, above, essentially also applies in this context. The proposed new inclusion tables [IDNA2008-Tables], the reduction in the number of characters permitted as input for registration or lookup (Section 5), and even the proposed changes in handling of right to left strings [IDNA2008-Bidi] either give interpretations to strings prohibited under IDNA2003 or prohibit strings that IDNA2003 permitted. Strings that are valid under both IDNA2003 and IDNA2008, 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 [IDNA2008-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.

9.2.5. The Symbol Question

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

- 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.

- Most Unicode names for letters are, in most cases, fairly intuitive, unambiguous and recognizable to users of the relevant script. 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 just one example, consider a "heart" symbol as it might appear in a logo that might be read as "I love...". While the user might read such a logo as "I love..." or "I heart...", considerable knowledge of the coding distinctions made in Unicode is needed to know that there more than one "heart" character (e.g., U+2665, U+2661, and U+2765) and how to describe it. These issues are of particular importance if strings are expected to be understood or transcribed by the listener after being read out loud. [[anchor33: The above paragraph remains controversial as to whether it is valid. The WG will need to make a decision if this section is not dropped entirely.]]

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

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

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

[[anchor34: Note in Draft: Should the above section be significantly trimmed or eliminated?]]

9.2.6. Migration Between Unicode Versions: Unassigned Code Points

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

In this specification, strings containing unassigned code points MUST NOT be either looked up or registered. There are several reasons for this, with the most important ones being:

- It cannot be known with sufficient reliability in advance that a code point that was not previously assigned will not be assigned to a compatibility character. In IDNA2003, since there is no direct dependency on NFKC (Stringprep's tables are based on NFKC, but IDNA2003 depends only on Stringprep), allocation of a compatibility character might produce some odd situations, but it would not be a problem. In IDNA2008, where compatibility characters are generally assigned to DISALLOWED, permitting strings containing unassigned characters to be looked up would permit violating the principle that characters in DISALLOWED are not looked up.

- More generally, the status of an unassigned character with regard to the DISALLOWED and PROTOCOL-VALID categories, and whether contextual rules are required with the latter, cannot be evaluated until a character is actually assigned and known.

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

9.2.7. Other Compatibility Issues

The existing (2003) IDNA model includes several odd artifacts of the context in which it was developed. Many, if not all, of these are potential avenues for exploits, especially if the registration process permits "source" names (names that have not been processed through IDNA and nameprep) to be registered. As one example, since the character Eszett, used in German, is mapped by IDNA2003 into the sequence "ss" rather than being retained as itself or prohibited, a string containing that character but that is otherwise in ASCII is not really an IDN (in the U-label sense defined above) at all. After Nameprep maps the Eszett out, the result is an ASCII string and so does not get an xn-- prefix, but the string that can be displayed to a user appears to be an IDN. The proposed IDNA2008 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.

10. Acknowledgments

The editor and contributors would like to express their thanks to those who contributed significant early (pre-WG) review comments, sometimes accompanied by text, 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, Erik van der Poel, 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.

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

Special thanks are due to Paul Hoffman for permission to extract material from his Internet-Draft to form the basis for Section 9.1.

Useful comments and text on the WG versions of the draft were received from many participants in the IETF "IDNABIS" WG and a number of document changes resulted from mailing list discussions made by that group. Marcos Sanz provided specific analysis and suggestions that were exceptionally helpful in refining the text.

11. Contributors

While the listed editor held the pen, this core of this document and the initial WG version represents the joint work and conclusions of an ad hoc design team consisting of the editor and, in alphabetic order, Harald Alvestrand, Tina Dam, Patrik Faltstrom, and Cary Karp. 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.

12. Internationalization Considerations

DNS labels and fully-qualified domain names provide mnemonics that assist in identifying and referring to resources on the Internet. IDNs expand the range of those mnemonics to include those based on languages and character sets other than Western European and Roman-

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

13. IANA Considerations

This section gives an overview of registries required for IDNA. The actual definitions of the first two appear in [IDNA2008-Tables].

13.1. IDNA Character Registry

The distinction among the three major categories "UNASSIGNED", "DISALLOWED", and "PROTOCOL-VALID" is made by special categories and rules that are integral elements of [IDNA2008-Tables]. Convenience in programming and validation requires a registry of characters and

scripts and their categories, updated for each new version of Unicode and the characters it contains. The details of this registry are specified in [IDNA2008-Tables].

13.2. IDNA Context Registry

For characters that are defined in the IDNA Character Registry list as PROTOCOL-VALID but requiring a contextual rule (i.e., the types of rule described in Section 5.1.1.1), IANA will create and maintain a list of approved contextual rules. The details for those rules appear in [IDNA2008-Tables].

13.3. IANA Repository of IDN Practices of TLDs

This registry, historically described as the "IANA Language Character Set Registry" or "IANA Script Registry" (both somewhat misleading terms) is maintained by IANA at the request of ICANN. It is used to provide a central documentation repository of the IDN policies used by top level domain (TLD) registries who volunteer to contribute to it and is used in conjunction with ICANN Guidelines for IDN use.

It is not an IETF-managed registry and, while the protocol changes specified here may call for some revisions to the tables, these

specifications have no direct effect on that registry and no IANA action is required as a result.

14. 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 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 administrators 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 suggestions about identification and handling of confusable characters appear in a Unicode Consortium publication [Unicode-UTR36].

The registration and lookup models described above and in [IDNA2008-Protocol] change the mechanisms available for lookup applications 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 9.2.3.2) 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.

No mechanism involving names or identifiers alone can protect a wide variety of security threats and attacks that are largely independent of them including spoofed pages, DNS query trapping and diversion, and so on.

15. Change Log

[[anchor42: RFC Editor: Please remove this section.]]

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

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

- Rewrote the discussion of Labels in Registration (Section 9.2.1.2) and related text in Section 1.5.4.1.1 to narrow its focus and remove more general restrictions. Added a temporary note in line to explain the situation.

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

controversial assertions.

- Added a discussion of examples to Section 9.2.1

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

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

15.2. Version -02

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

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

- Rewrote the material on preprocessing somewhat.

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

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

16. References

16.1. Normative References

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

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

[IDNA2008-Bidi] Alvestrand, H. and C. Karp, "An updated IDNA criterion for right to left scripts", July 2008, <https://datatracker.ietf.org/drafts/draft-ietf-idnabs-bidi/>.

[IDNA2008-Protocol] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", July 2008, <https://datatracker.ietf.org/drafts/draft-ietf-idnabis-protocol/>.

[IDNA2008-Tables]
Faltstrom, P., "The Unicode Code Points and IDNA", July 2008, <https://datatracker.ietf.org/drafts/draft-ietf-idnabis-tables/>.

A version of this document is available in HTML format at http://stupid.domain.name/idnabis/draft-ietf-idnabis-tables-02.html

[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.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RulesInit]
Klensin, J., "Internationalizing Domain Names in Applications (IDNA): Protocol, Appendix A Contextual Rules Table", July 2008, <http://www.ietf.org/internet-drafts/draft-ietf-idnabis-protocol-02.txt>.

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

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

defined by: The Unicode Standard, Version 5.0, Boston, MA, Addison-Wesley, 2007, ISBN 0-321-48091-0, as amended by Unicode 5.1.0 (http://www.unicode.org/versions/Unicode5.1.0/).

16.2. Informative References

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

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

[GB18030] "Chinese National Standard GB 18030-2000: Information Technology -- Chinese ideograms coded character set for information interchange -- Extension for the basic set.", 2000.

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

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

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

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

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

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

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

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

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

[RFC3743] Konishi, K., Huang, K., Qian, H., and Y. Ko, "Joint Engineering Team (JET) Guidelines for Internationalized Domain Names (IDN) Registration and Administration for Chinese, Japanese, and Korean", RFC 3743, April 2004.

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

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

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

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

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

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

Author's Address

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

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

Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be

found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.