Network Working Group
Request for Comments: 4290
Category: Informational

J. Klensin

December 2005

Suggested Practices for Registration of Internationalized Domain Names (IDN)

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

IESG Note

This RFC is not a candidate for any level of Internet Standard. The IETF disclaims any knowledge of the fitness of this RFC for any purpose and notes that the decision to publish is not based on IETF review apart from IESG review for conflict with IETF work. The RFC Editor has chosen to publish this document at its discretion. See RFC 3932 for more information.

Abstract

This document explores the issues in the registration of internationalized domain names (IDNs). The basic IDN definition allows a very large number of possible characters in domain names, and this richness may lead to serious user confusion about similar-looking names. To avoid this confusion, the IDN registration process must impose rules that disallow some otherwise-valid name combinations. This document suggests a set of mechanisms that registries might use to define and implement such rules for a broad range of languages, including adaptation of methods developed for Chinese, Japanese, and Korean domain names.

Table of Contents

1. Introduction

1.1. Background

The IDNA (Internationalized Domain Names in Applications) specification [RFC3490] defines the basic model for encoding non-ASCII strings in the DNS. Additional specifications [RFC3491] [RFC3492] define the mechanisms and tables needed to support IDNA. As work on these specifications neared completion, it became apparent that it would be desirable for registries to impose additional restrictions on the names that could actually be registered (e.g., see [IESG-IDN] and [ICANN-IDN]) to reduce potential confusion among characters that were similar in some way. This document explores these IDN (international domain name) registration issues and suggests a set of mechanisms that IDN registries might use.

Registration restrictions are part of a long tradition. For example, while the original DNS specifications [RFC1035] permitted any string of octets in a DNS label, they also recommended the use of a much more restricted subset. This subset was derived from the much older "hostname" rules [RFC952] and defined by the "LDH" convention (for the three permitted types of characters: letters, digits, and the hyphen). Enforcement of this restricted subset in registrations was the responsibility of the registry or domain administrator. The definition of the subset was embedded in the DNS protocol itself, although some applications protocols, notably those concerned with electronic mail, did impose and enforce similar rules.

If there are no constraints on registration in a zone, people can register characters that increase the risk of misunderstandings, cybersquatting, and other forms of confusion. A similar situation existed even before the introduction of IDNA, as exemplified by domain names such as example.com and examp1e.com (note that the latter domain contains the digit "1" instead of the letter "l").

For non-ASCII names (so-called "internationalized domain names" or "IDNs"), the problem is more complicated. In the earlier situation that led to the LDH (hostname) rules, all protocols, hosts, and DNS zones used ASCII exclusively in practice, so the LDH restriction could reasonably be applied uniformly across the Internet. Support for IDNs introduces a very large character repertoire, different geographical and political locations, and languages that require different collections of characters. The optimal registration restrictions are no longer a global matter; they may be different in different areas and, hence, in different DNS zones.

For some human writing systems, there are characters and/or strings that have equivalent or near-equivalent usages. If a name can be registered with such a character or string, the registry might want to automatically associate all of the names that have the same meaning with the registered name. The registry might also decide whether the names that are associated with, or generated by, one registration should, as a group or individually, go into the zone or should be blocked from registration by different parties.

To date, the best-developed system for handling registration restrictions for IDNs is the JET Guidelines for Chinese, Japanese, and Korean [RFC3743], the so-called "CJK" languages. The JET Guidelines are limited to the CJK languages and, in particular, to their common script base. Those languages are also the best-known and most widely-used examples of writing systems constructed on "ideographic" or "pictographic" principles. This document explores the principles behind the JET guidelines. It then examines some of the issues that might arise in adapting them to alphabetic languages, i.e., to languages whose characters primarily represent sounds rather than meanings.

This document describes five things:

1. The general background and considerations for non-ASCII scripts in names.

2. Suggested practices for describing character variants.

3. A method for using a zone's character variants to determine which names should be associated with a registration.

4. A format for publishing a zone's table of character variants; Such tables are referred to below simply as "language tables" or simply "tables".

5. A model algorithm for name registration given the presence of language tables.

1.2. The Nature and Status of these Recommendations

The document makes recommendations for consideration by registries and, where relevant, by those who coordinate them, and by those who use their services. None of the recommendations are intended to be normative. Instead, the intent of the document is to illustrate a framework for developing variations to meet the needs of particular registries and their processing of particular languages. Of course, if registries make similar decisions and utilize similar tools, costs

and confusion may be reduced -- both between registries and for users and registrars who have relationships with more than one domain.

Just as the JET Guidelines contain some suggestions that may not be applicable to alphabetic scripts, some of the suggestions here, especially the more specific ones, may be applicable to some scripts and not others.

1.3. Terminology

1.3.1. Languages and Scripts

This document uses the term "language" in what may be, to many readers, an odd way. Neither this specification, nor IDNA, nor the DNS are directly concerned with natural language, but only with the characters that make up a given label. In some respects, the term "script", used in the character coding community for a collection of characters, might be more appropriate. However, different subsets of the same script may be used with different languages, and the same language may be written using different characters (or even completely different scripts) in different locations, so "script" is not precisely correct either.

Long-standing confusion has also resulted from the fact that most scripts are, informally at least, named after one of the languages written in them. "Chinese" describes both a language and a collection of characters that are also used in writing Japanese, Korean, and, at least historically, some other languages. "Latin" describes a language, the characters used to write that language, and, often, characters used to write a number of contemporary languages that are derived from or similar to those used to write the Latin language. The script used to write the Arabic language is called "Arabic", but it is also used (typically with some additions or deletions) to write a number of other languages. Situations in which a script has a clearly-defined name that is independent of the name of a language are the exception, rather than the rule; examples include Hangul, used to write Korean, Katakana and Hiragana, used to write Japanese, and a few others. Some scholars have historically used "Roman" or "Roman-derived" for the script in an attempt to distinguish between a script and the Latin language.

The term "language" is therefore used in this document in the informal sense of a written language and is defined, for this purpose, by the characters used to write it, i.e., as a language-specific subset of a script. In this context, a "language" is defined by the combination of a code (see Section 1.4.1) and an authority that has chosen to use that code and establish a character-listing for it. Authorities are normally TLD (top-level

domain) registries; see Section 7 and [IANA-language-registry]. However, it is expected that TLD registries will find appropriate experts and that advice from language and script experts selected by international neutral bodies will also become part of the registration system. In addition, as discussed below in Section 7, registries may conclude that the best interests of registrants, stakeholders, and the Internet community would be served by constructing "language tables" that mix scripts and characters in ways that conform to no known language. Conventions should be developed for such registrations that do not misleadingly reflect specific language codes.

1.3.2. Characters, Variants, Registrations, and Other Issues

1. Characters in this document are specified by their Unicode codepoints in U+xxxx format, by their official names, or both.

2. The following terms are used in this document.

   * String

     A "string" is an sequence of one or more characters.

   * Base Character

     This document discusses characters that may have equivalent or near-equivalent characters or strings. A "base character" is a character that has zero or more equivalents. In the JET Guidelines, base characters are referred to as "valid characters". In a table with variants, as described in Section 5, the base characters occupy the first column. Normally (and always, if the recommendation of Section 6.3 is adopted), the base characters will be the characters that appear in registration requests from registrants; any other character will invalidate the registration attempt.

   * Native Script

     Native script is the form in which the relevant string would normally be represented. For example, it might use Lower Slobbovian characters and the glyphs normally used to write them. It would not be punycode as a presentation form.

   * Variant Characters/Strings

     The "variant(s)" are character(s) and/or string(s) that are treated as equivalent to the base character. Note that these might not be exactly equivalent characters; a particular

original character may be a base character with a mapping to a particular variant character, but that variant character may not have a mapping to the original base character. Indeed, the variant character may not appear in the base character list, and hence may not be valid for use in a registration. Usually, characters or strings to be designated as variants are considered either equivalent or sufficiently similar (by some registry-specific definition) that confusion between them and the base character might occur.

* Base Registration

  The "base registration" is the single name that the registrant requested from the registry. The JET Guidelines use the term "label string" for this concept.

* Registered, Activated

  A label (or "name") is described as "registered" if it is actually entered into a domain (i.e., into a zone file) by the registry, so that it can be accessed and resolved using standard DNS tools. The JET Guidelines describe a "registered" label as "activated". However, some domains use a slightly different registration logic in which a name can be registered with the registrar (if one is involved) and with the registry, but not actually entered into the zone file until an additional activation or delegation step occurs. This document does not make that distinction, but is compatible with it.

  As specified in the IDNA Standard, the name actually placed in the zone file is always the internal ("punycode") form. There is no provision for actually entering any other form of an IDN into the DNS. It remains controversial, with different registrars and registries having adopted different policies, as to whether the registration, as submitted by the registrant, is in the form of:

  o The native-script name, either in UTF-8 or in some coding specified by the registrar, or

  o the internal-form ("punycode") name, or

  o both forms of the name together, so that the registrar and registry can verify the intended translation.

If any of the approaches defined in this document is used, it is almost certain to be necessary that the native-script form of the requested string be available to the registry.

* Registration Bundle

  A "registration bundle" is the set of all labels that come from expanding the base characters for a single name into their variants. The presence of a label in a registration bundle does not imply that it is registered. In the JET Guidelines, a registration bundle is called an "IDN Package".

* Reserved Label

  A "reserved label" is a label in a registration bundle that is not actually registered.

* Registry"

  A "registry" is the administrative authority for a DNS zone. The registry is the body that enforces, and typically makes, policies that are used in a particular zone in the DNS.

* Coded Character Set

  A "Coded Character Set" (CCS) is a list of characters and the code positions assigned to them. ASCII and Unicode are CCSs.

* Language

  A "language" is something spoken by humans, independent of how it is written or coded. ISO Standard 639 and IETF BCP 47 (RFC 3066) [RFC3066] list and define codes for identifying languages.

* Script

  A "script" is a collection of characters (glyphs, independent of coding) that are used together, typically to represent one or more languages. Note that the script for one language may heavily overlap the script for another. This does not imply that they have identical scripts.

* Charset

  "Charset" is an IETF-invented term to describe, more or less, the combination of a script, a CCS that encodes that script,

and rules for serializing encoded bytes that are stored on a computer or transmitted over the network.

The last four of these definitions are redundant with, but deliberately somewhat less precise than, the definitions in [RFC3536], which also provides sources. The two sets of definitions are intended to be consistent.

1.3.3. Confusion, Fraud, and Cybersquatting

The term "confusion" is used very generically in this document to cover the entire range from accidental user misperception of the relationship between characters with some characteristic in common (typically appearance, sound, or meaning) to cybersquatting and (other) deliberately fraudulent attempts to exploit those relationships based on the nature of the characters.

1.4. A Review of the JET Guidelines

1.4.1. JET Model

In the JET Guidelines model, a prospective registrant approaches the registry for a zone (perhaps through an intermediate registrar) with a candidate base registration -- a proposed name to be registered -- and a list of languages in which that name is to be interpreted. The languages are defined according to the fairly high-resolution coding of [RFC3066] or, if the registry considers it more appropriate, a coding based on scripts such as those in [LTRU-Registry]. In this way, Chinese as used on the mainland of the People's Republic of China ("zh-cn") can, at registry option, consist of a somewhat different list of characters (code points) and be represented by a separate table compared to Chinese as used in Taiwan ("zh-tw").

The design of the JET Guidelines took one important constraint as a basis: IDNA was treated as a firm standard. A procedure that modified some portion of the IDNA functions, or was a variant on them, was considered a violation of those standards and should not be encouraged (or, probably, even permitted).

Each registry is expected to construct (or obtain) a table for each language it considers relevant and appropriate. These tables list, for the particular zone, the characters permitted for that language. If a character does not appear as a base character (called a "valid code point" in the JET document) in that table, then a name containing it cannot be registered. If multiple languages are listed for the registration, then the character must appear in the tables for each of those languages.

The tables may also contain columns that specify alternate or variant forms of the valid character. If these variants appear, they are used to synthesize labels that are alternatives to the original one. These labels are all reserved and can be registered or "activated" (placed into the DNS) only by the action or request of the original registrant; some (the "preferred variant labels") are typically registered automatically. The zone is expected to establish appropriate policies for situations in which the variant forms of one label conflict with already-reserved or already-registered labels.

Most of these concepts were introduced because of concerns about specific issues with CJK characters, beginning from the requirement that the use of Simplified Chinese by some registrants and Traditional Chinese by others not be permitted to create confusion or opportunities for fraud. While they may be applicable to registry tables constructed for alphabetic scripts, the translation should be done with care, since many analogies are not exact.

Some of the important issues are discussed in the sections that follow, especially Section 3. The JET model may be considered as a variation on, and inspiration for, the model and method presented by the rest of this document, although the JET model has been completely developed only for CJK characters. Other languages or scripts, especially alphabetic ones, may require other variations.

1.4.2. Reserved Names and Label Packages

A basic assumption of the JET model is that, if the evolution of specific characters or the properties of Unicode [Unicode] [Unicode32] or IDNA cause two strings to appear similar enough to cause confusion, then both should be registered by the same party or one of them should become unregisterable. The definition of "appear similar enough" will differ for different cultures and circumstance, and hence DNS zones, but the principle is fairly general. In the JET model, all of the variant strings are identified, some are registered into the DNS automatically, and others are simply reserved and can be registered, if at all, only by the original registrant. Other zones might find other policies appropriate. For example, a zone might conclude that having similar strings registered in the DNS was undesirable. If so, the list of variant strings would be used only to build a list of names that would be reserved and prohibited from being registered.

1.5. Languages, Scripts, and Variants

1.5.1. Languages versus Scripts

Conversations about scripts -- collections of characters associated with particular languages -- are common when discussing character sets and codes. However, the boundaries between one script and another are not well-defined. The Unicode Standard ([Unicode], [Unicode32]), for example, does not define script boundaries at all, even though it is structured in terms of usually-related blocks of characters. The issue is complicated by the common origin of most alphabetic scripts in use in the world today (see, for example, [Drucker] or the more scholarly [Daniels]).

Because of that history, certain characters (or, more precisely, symbols representing characters) appear in the scripts associated with multiple languages, sometimes with very different sounds or meanings. This differs from the CJK situation in which, if a character appears in more than one of the relevant languages, it will usually have the same interpretation in each one. For the subset of characters that actually are ideographs or pictographs, pronunciation is expected to vary widely while meaning is preserved. At least in part because of that similarity of meaning, it made sense in the JET case to permit a registration to specify multiple languages, to verify that the characters in the label string (the requested "Base registration") were valid for each, and then to generate variant labels using each language in turn. For many alphabetic languages, it may be more sensible to prohibit the label string submitted for registration from being associated with more than one language. Indeed, "one label, one language" has been suggested as an important barrier against common sources of "look-alike" confusion. For example, the imposition of that rule in a zone would prevent the insertion of a few Greek or Cyrillic characters with shapes identical to the Latin ones into what was otherwise a Latin-based string. For a particular table, the list of base characters may be thought of as the script associated with the relevant language, with the understanding that the table design does not prevent the same character from appearing in the tables for multiple languages.

Indeed, this notion of a script that is local and specifically identified can be turned around: so-called "language tables" are associated with languages only insofar as thinking about the character structure and word forms associated with a given language helps to inform the construction of the table. A country like Finland, for example, might select among:

- One table each for Finnish, Swedish, and English characters and conventions, permitting a string to be registered in one, two, or

all three languages. However, a three-language registration would necessarily prohibit any characters that did not appear in all three languages, since the label would make little sense otherwise.

- One table each, but with a "one label, one language" rule for the zone.

- A combined table based on the observation that all three writing systems were based on Roman characters and that the possibilities for confusion of interest to the registry would not be reduced by "language" differentiation. This option raises an interesting issue about language labeling as described in Section 1.4.1; see the discussion in Section 7 below.

Regardless of what decisions were made about those languages and scripts, they might have a separate table for registration of labels containing Cyrillic characters. That table might contain some Roman-derived characters (either as base characters or as variants), just as some CJK tables do. See also Section 2, below.

Tables that present multiple languages, as described above, have introduced confusion and discomfort among those who have failed to understand these definitions. The consequence of these definitions is that use of a language or script code in a registration is a mnemonic, rather than a normative statement about the language or script itself. When that confusion is likely to occur, it is appropriate to simply use the registry identifier and a sequence number to identify the registration.

As the JET Guidelines stress, no tables or systems of this type -- even if identified with a language as a means of defining or describing the table -- can assure linguistic or even syntactic correctness of labels with regard to that language. That assurance may not be possible without human intervention or at least dictionary lookups of complete proposed labels. It may even not be desirable to attempt that level of correctness (see Section 2).

Of course, if any language-based tests or constraints, including "one label, one language", are to be applied to limit the associated sources of confusion, each zone must have a table for each language in which it expects to accept registrations. The notion of a single combined table for the zone is, in the general case, simply unworkable. One could use a single table for the zone if the intent were to impose only minimal restrictions, e.g., to force alphabetic and numeric characters only, excluding symbols and punctuation. That type of restriction might be useful in eliminating some problems, such as those of unreadable labels, but it would be unlikely to be

very helpful with, e.g., confusion caused by similar-looking characters.

1.5.2. Variant Selection

The area of character variants is rife with difficulties (and perhaps opportunities). There is no universal agreement about which base characters have variants, or if they do, what those variants are. For example, in some regions of the world and in some languages, LATIN SMALL LETTER O WITH DIAERESIS (U+00F6) and LATIN SMALL LETTER O WITH STROKE (U+00F8) are variants of each other, while in other regions, most people would think that LATIN SMALL LETTER O WITH STROKE has no variants. In some cases, the list of variants is difficult to enumerate. For example, it required several years for the Chinese language community to create variant tables for use with IDNA, and it remains, at the time of this writing, questionable how widely those tables will be accepted among users of Chinese from areas of the world other than those represented by the groups that created them.

Thus, the first thing a registry should ask is whether or not any of the characters that they want to permit to be used have variants. If not, the registry's work is much simpler. This is not to say that a registry should ignore variants if they exist: adding variants after a registry has started to take registrations will be nearly as difficult administratively as removing characters from the list of acceptable characters. That is, if a registry later decides that two characters are variants of each other, and there are actively-used names in the zones that differ only on the new variants, the registry might have to transfer ownership of one of the names to a different owner, using some process that is certain to be controversial.

This situation in likely to be much easier for areas and zones that use characters that previously did not occur in the DNS at all than it will be for zones in which non-English labels have been registered in ASCII characters for some time, presumably because the language of interest uses additional "Latin" characters with some conventions when only ASCII is available. In the former case, the rules and conventions can be established before any registrations occur. In the latter, there may be conflicts or opportunities for confusion between existing registrations and now-permitted Roman-based characters that do not appear in ASCII. For example, a domain name might exist today that uses the name of a city in Canada spelled as "Montreal". If the zone in which it occurs changes its rules to permit the use of the character LATIN SMALL LETTER E WITH ACUTE (U+00E9), does the name of the city, spelled (correctly) using that character, conflict with the existing domain name registration?

Certainly, if both are permitted, and permitted to be registered by separate parties, there are many opportunities for confusion.

Of course, zone managers should inform all current registrants when the registration policy for the zone changes. This includes the times when IDN characters are first allowed in the zone, when additional characters are permitted, and when any change occurs in the character variant tables.

Many languages contain two variants for a character, one of which is strongly preferred. A registry might restrict the base registration to the preferred form, or it might allow any form for the base registration. If the variant tables are created carefully, the resulting bundles will be the same, but some registries will give special status to the base registration such as its appearance in "Whois" databases.

1.6. Variants are not a Universal Remedy

It is worth stressing that there are many obvious opportunities for confusion that variant systems, by virtue of being based on processing of individual characters, cannot address. For example, if a language can be written with more than one script, or transliterations of the language into another script are common, variant models are insufficient to prevent conflicting registration of the related forms. Avoiding those types of problems would require different mechanisms, perhaps based on phonetic or natural language processing techniques for the entire proposed base registration.

1.7. Reservations and Exclusions

1.7.1. Sequence Exclusions for Valid Characters

The JET Guidelines are based on processing only single characters. Pairs or longer sequences of characters can, at the option of the registry, be handled through what the Guidelines describe as "additional processing". These registry-specific string processing procedures are specifically permitted by the guidelines to supplement the per-character processing that generates the variants.

A different zone with different needs could use a modified version of the table structure, or different types of additional processing, to prohibit particular sequences of characters by marking them as invalid, and to accept characters by marking them as valid. Other modifications or extensions might be designed to prevent certain letters from appearing at the beginning or end of labels. The use of regular expressions in the "valid characters" column might be one way

to implement these types of restrictions, but there has been no experience so far with that approach.

In particular, in some scripts derived from Roman characters, sequences that have historically been typographically represented by single "ligature" or "digraph" characters may also be represented by the separate characters (e.g., "ae" for U+00E6 or "ij" for U+0133). If it is desired to either prohibit these, or to treat them as variants, some extensions to the single-character JET model may be needed. Some careful thinking about IDNA (especially nameprep) may also be needed, since some of these combinations are excluded there).

1.7.2. Character Pairing Issues

Some character pairings -- the use of a character form (glyph) in one language and a different form with the same properties in a related one -- closely approximate the issues with mapping between Traditional and Simplified Chinese, although the history is different. For example, it might be useful to have "o" with a stroke (U+00F8) as a variant for "o" with diaeresis above it (U+00F6) (and the equivalent upper-case pair) in a Swedish table, and vice versa in a Norwegian one, or to prohibit one of these characters entirely in each table. In a German table, U+00F8 would presumably be prohibited, while U+00F6 might have "oe" as a variant. Obviously, if the relevant language of registration is unknown, this type of variant matching cannot be applied in any sensible way.

1.8. The Registration Bundle

1.8.1. Definitions and Structure

As one of its critical innovations, the JET model defines an "IDN package", known in this document as a "registration bundle", which consists of the primary registered string (which is used as the name of the bundle), the information about the language table(s) used, the variant labels for that string, and indications of which of those labels are registered in the relevant zone file ("activated" in the JET terminology). Registration bundles are also atomic -- one can not add or remove variant labels from one without unregistering the entire package. A label exists in only one registration bundle at a time; if a new label is registered that would generate a variant that matches one that appears in an existing package, that variant simply is not included in the second package. A subsequent de-registration of the first package does not cause the variant to be added to the second. While it might be possible to change this in other models, the JET conclusion was that other options would be far too complex to implement and operate and would cause many new types of name conflicts.

1.8.2. Application of the Registration Bundle

A registry has three options for handling the case where the registration bundle contains more than one label. The policy options are:

- Register and resolve all labels in the zone, making the zone information identical to that of the registered labels. This option will allow end users to find names with variants more easily, but will result in larger zone files. For some language tables, the zone file could become so large that it could negatively affect the ability of the registry to perform name resolution. If the base registration contains several characters that have equivalents, the owner could end up having to take care of large numbers of zones. For instance, if DIGIT ONE is a variant of LATIN SMALL LETTER L, the owner of the domain name all-lollypops.example.com will have to manage 32 zones. If the intent is to keep the contents of those zones identical, the owner may then face a significant administrative problem. If other concerns dictate short times to live and absolute consistency of DNS responses, the challenges may be nearly impossible.

- Block all labels other than the registered label so they cannot be registered in the future. This option does not increase the size of the zone file and provides maximum safety against false positives, but it may cause end users to not be able to find names with variants that they would expect. If the base registration contains characters that have equivalents, Internet users who do not know what base characters were used in the registration will not know what character to type in to get a DNS response. For instance, if DIGIT ONE is a variant of LATIN SMALL LETTER L, and LATIN SMALL LETTER L is a variant of DIGIT ONE, the user who sees "pale.example.com" will not know whether to type a "1" or a "l" after the "pa" in the first label.

- Resolve some labels and block some other labels. This option is likely to cause the most confusion with users because including some variants will cause a name to be found, but using other variants will cause the name to be not found. For example, even if people understood that DIGIT ONE and LATIN SMALL LETTER L were variants, a typical DNS user wouldn't know which character to type because they wouldn't know whether this pair were used to register or block the labels. However, this option can be used to balance the desires of the name owner (that every possible attempt to enter their name will work) with the desires of the zone administrator (to make the zone more manageable and possibly to be compensated for greater amounts of work needed for a single

registration). For many circumstances, it may be the most attractive option.

In all cases, at least the registered label should appear in the zone. It would be almost impossible to describe to name owners why the name that they asked for is not in the zone, but some other name that they now control is. By implication, if the requested label is already registered, the entire registration request must be rejected.

2. Some Implications of This Approach

Historically, DNS labels were considered to be arbitrary identifier strings, without any inherent meaning. Even in ASCII, there was no requirement that labels form words. Labels that could not possibly represent words in any Romance or Germanic language (the languages that have been written in "Latin" scripts since medieval times or earlier) have actually been quite common. In general, in those languages, words contain at least one vowel and do not have embedded numbers. As a result, a string such as "bc345df" cannot possibly be a "word" in these languages. More generally, the more one moves toward "language"-based registry restrictions, the less it is going to be possible to construct labels out of fanciful strings. While fanciful strings are terrible candidates for "words", they may make very good identifiers. To take a trivial example using only ASCII characters, "rtr32w", "rtr32x", and "rtr32z" might be very good DNS labels for a particular zone and application. However, given the embedded digits and lack of vowels, they, like the "bc345df" example given above, would fail even the most superficial of tests for valid English (or German or French (etc.)) word forms.

It is worth noting that several DNS experts have suggested that a number of problems could be solved by prohibiting meaningful names in labels, requiring instead that the labels be random or nonsense strings. If methods similar to those discussed in this document were used to force identifiers to be closer to meaningful words in real languages, the result would be directly contradictory to those "random name" approaches.

Interestingly, if one were trying to develop an "only words" system, a rather different -- but very restrictive -- model could be developed using lookups in a dictionary for the relevant language and a listing of valid business names for the relevant area. If a string did not appear in either, it would not be permitted to be registered. Models that require a prior national business listing (or registration) that is identical to the proposed domain name label have historically been used to restrict registrations in some country-code top level domains, so this is not a new idea. On the other hand, if look-alike characters are a concern, even that type of

rule (or restriction) would still not avoid the need to consider character variants.

Consequently, registries applying the principles outlined in this document should be careful not to apply more severe restrictions than are reasonable and appropriate while, at the same time, being aware of how difficult it usually is to add restrictions at a later time.

3. Possible Modifications of the JET Model

The JET model was designed for CJK characters. The discussion above implies that some extensions to it may be needed to handle the characteristics of various alphabetic scripts and the decisions that might be made about them in different zones. Those extensions might include facilities to process:

- Two-character (or more) sequences, such as ligatures and typographic spelling conventions, as variants.

- Regular expressions or some other mechanism for dealing with string positions of characters (e.g., characters that must, or must not, appear at the beginning or end of strings).

- Delimiter breaks to permit multiple languages to be used, separately, within the same label. E.g., is it possible to define a label as consisting of two or more components, each in a different language, with some particular delimiter to define the boundaries of the components?

4. Conclusions and Recommendations About the General Approach

After examining the implications of the potential use of the full range of characters permitted by IDNA in DNS labels, multiple groups, including IESG [IESG-IDN] and ICANN [ICANN-IDN] [ICANN-IDN2], have concluded that some restrictions are needed to prevent many forms of user confusion about the actual structure of a name or the word, phrase, or term that it appears to spell out. The best way to approach such restrictions appears to draw from the language and culture of the community of registrants and users in the relevant zone: if particular characters are likely to be surprising or unintelligible to both of those groups, it is probably wise to not permit them to be used in registrations. Registration restrictions can be carried much further than restricting permitted characters to a selected Unicode subset. The idea of a reserved "bundle" of related labels permits probably-confusing combinations or sets of characters to be bound together, under the control of a single registrant. While that registrant might still use the package in a way that confused his or her own users (the approach outlined here

will not prevent either ill-though-out ideas or stupidity), the possibility of turning potential confusion into a hostile attack would be considerably reduced.

At the same time, excessive restrictions may make DNS identifiers less useful for their original purpose: identifying particular hosts and similar resources on the network in an orderly way. Registries creating rules and policies about what can be registered in particular zones -- whether those are based on the JET Guidelines or the suggestions in this document -- should balance the need for restrictions against the need for flexibility in constructing identifiers.

The discussion above provides many options that could be selected, defined, and applied in different ways in different registries (zones). Registrars and registrants would almost certainly prefer systems in which they can predict, at least to a first order approximation, the implications of a particular potential registration. Predictability of that sort probably requires more standards, and less flexibility, than the model itself might suggest.

5. A Model Table Format

The format of the table is meant to be machine-readable but not human-readable. It is fairly trivial to convert the table into one that can be read by people.

Each character in the table is given in the "U+" notation for Unicode characters. The lines of the table are terminated with either a carriage return character (ASCII 0x0D), a linefeed character (ASCII 0x0A), or a sequence of carriage return followed by linefeed (ASCII 0x0D 0x0A). The order of the lines in the table may or may not matter, depending on how the table is constructed.

Comment lines in the table are preceded with a "#" character (ASCII 0x2C).

Each non-comment line in the table starts with the character that is allowed in the registry and expected to be used in registrations, which is also called the "base character". If the base character has any variants, the base character is followed by a vertical bar character ("|", ASCII 0x7C) and the variant string. If the base character has more than one variant, the variants are separated by a colon (":", ASCII 0x3A). Strings are given with a hyphen ("-", ASCII 0x2D) between each character. Comments beginning with a "#" (ASCII 0x2C), and may be preceded by spaces (" ", ASCII 0x20).

The following is an example of how a table might look. The entries in this table are purposely silly and should not be used by any registry as the basis for choosing variants. For the example, assume that the registry:

- allows the FOR ALL character (U+2200) with no variants
- allows the COMPLEMENT character (U+2201) which has a single variant of LATIN CAPITAL LETTER C (U+0043)
- allows the PROPORTION character (U+2237) which has one variant which is the string COLON (U+003A) COLON (U+003A)
- allows the PARTIAL DIFFERENTIAL character (U+2202) which has two variants: LATIN SMALL LETTER D (U+0064) and GREEK SMALL LETTER DELTA (U+03B4)

The table contents (after any required header information, see [IANA-language-registry] and the discussion in Section 7 below) would look like:

```
# An example of a table
U+2200
U+2201|U+0043
U+2237|U+003A-U+003A # Note that the variant is a string
U+2202|U+0064:U+03B4 # Two variants for the same character
```

Implementers of table processors should remember that there are tens of thousands of characters whose codepoints are greater than 0xFFFF. Thus, any program that assumes that each character in the table is represented in exactly six octets ("U", "+", and four octets representing the character value) will fail with tables that use characters whose value is greater than 0xFFFF.

6. A Model Label Registration Procedure: "CreateBundle"

This procedure has three inputs:

1. the proposed base registration,
2. the language (or script, if the registration is script-based, but "language" is used for convenience below) for the proposed base registration, and
3. the processing table associated with that language.

The output of the process is either failure (the base registration cannot be registered at all), or a registration bundle that contains

one or more labels (always including the base registration). As described earlier, the registration bundle should be stored with its date of creation so that issues with overlapping elements between bundles can later be resolved on a first-come, first-served basis.

There are two steps to processing the registration:

1. Check whether the proposed base registration exists in any bundle. If it does, stop immediately with a failure.

2. Process the base registration with the mechanism described as "CreateBundle" in Section 6.1, below.

Note that the process must be executed only once. The process must not be performed on any output of the process, only on the proposed base registration.

6.1. Description of the CreateBundle Mechanism

The CreateBundle mechanism determines whether a registration bundle can be created and, if so, populates that bundle with valid labels.

During the processing, a "temporary bundle" contains partial labels, that is, labels that are being built and are not complete labels. The partial labels in the temporary bundle consist of strings.

The steps are:

1. Split the base registration into individual characters, called "candidate characters". Compare every candidate character against the base characters in the table. If any candidate character does not exist in the set of base characters, the system must stop and not register any names (that is, it must not register either the base registration or any labels that would have come from character variants).

2. Perform the steps in IDNA's ToASCII sequence for the base registration. If ToASCII fails for the base registration, the system must stop and not register any label (that is, it must not register either the base registration or labels that might have been created from variants of characters contained in it). If ToASCII succeeds, place the base registration into the registration bundle.

3. For every candidate character in the base registration, do the following:

- Create the set of characters that consists of the candidate character and any variants.

- For each character in the set from the previous step, duplicate the temporary bundle that resulted from the previous candidate character, and add the new character to the end of each partial label.

4. The temporary bundle now contains zero or more labels that consist of Unicode characters. For every label in the temporary bundle, do the following:

   - Process the label with ToASCII to see if ToASCII succeeds. If it does, add the label to the registration bundle. Otherwise, do not process this label from the temporary bundle any further; it will not go into the registration bundle.

The result of the processing outlined above is the registration bundle with the base registration and possibly other labels.

6.2. The "no-variants" Case

It is clear that, for many scripts, registries will choose to create tables without variants, either because variants are clearly not necessary or because they are determined to cause more confusion and overhead than is justified by the circumstances. For those situations the table model of Section 5 becomes a trivial listing of base characters and only the first two steps of CreateBundle (verifying that all candidate character are in the base ("valid") character list and verifying that the resulting characters will succeed in the ToASCII operation) are applicable. Even the second of those steps becomes pro forma if the advice in the next subsection is followed.

6.3. CreateBundle and Nameprep Mapping

One of the functions of Nameprep, and IDNA more generally, is to map a large number of Unicode characters (code points) into a smaller number to avoid a different but overlapping set of confusion problems. For example, when a non-ASCII script makes distinctions between "upper case" and "lower case", nameprep maps the upper case characters to the lower case ones in order to simulate the DNS protocol's rule that ASCII characters are interpreted in a case-insensitive way. Unicode also contains many code points that are typographic variants on each other (e.g., forms with different widths and code points that designate font variations for mathematical uses), the Unicode standard explicitly identifies them that way, and Nameprep maps these onto base characters.

While having these mapping functions available during lookup may be quite helpful to users who type equivalent forms, registrations are probably best performed in terms of the IDNA base characters only, i.e., those characters that nameprep will not change. This will have two advantages.

- Registrants will never find themselves in the rather confusing position of having submitted one string for registration and finding a different string in the registry database (which could otherwise occur even if the relevant language table does not contain variants).

- Those who are interested in what characters are permitted by a given registry will only need to examine the relevant tables, rather than simulating the IDNA algorithm to determine the result of processing particular characters.

7. IANA Considerations

Under ICANN (not IETF) direction and management, the IANA has created a registry for language variant tables. The authoritative documentation for that registry is in [IANA-language-registry]. Since the registry exists and is being managed under ICANN direction, the material that follows is a review of the theory of this registry, rather than new instructions for IANA.

As described above and suggested in the JET Guidelines, the registration rules generally require only that:

- The application be submitted or endorsed by a TLD registry, to ensure that someone cares about the particular table.

- The table be identified by the following:

  * the name -- usually the top-level domain name -- of the submitting or endorsing registry;

  * one of: a language designation (consistent with [RFC3066] or with some other system approved by the IANA), a script designation, a combination of the two, or a sequence number acceptable to IANA for this purpose;

  * a version number; and

  * a date.

- Characters listed in the table be identified by Unicode code points, as discussed above.

- The table format may correspond to that identified in [RFC3743], or in Section 5 above, or may be some variation on those themes appropriate to the local processing model (with or without variants).

This raises some issues that will need to be worked out as experiences accumulate. For example, more standardization of table formats would be desirable to allow processing by the same computer tools for different registries and languages. But standardization seems premature at this time due to differences in languages, processing, and requirements and lack of experience with them. Similarly, if a registry concludes that it should use a table that contains characters from several scripts, it is not clear how such a table should be designated. Identifying it with a language code (either according to [RFC3066] or an independent code registered with IANA) is likely to just introduce more confusion, especially given other Internet uses of the language codes. It appears that some other convention will be needed for those cases, and it should be developed (if it has not already been established by the time this document is published).

8. Internationalization Considerations

This document specifies a model mechanism for registering Internationalized Domain Names (IDNs) that can be used to reduce confusion among similar-appearing names. The proposal is designed to facilitate internationalization while permitting a balance between internationalization concerns and concerns about keeping the Internet global and domain name system references unique in the perception of the user as well as in practice.

9. Security Considerations

Registration of labels in the DNS that contain essentially unrestricted sequences of arbitrary Unicode characters may introduce opportunities for either attacks or simple confusion. Some of these risks, such as confusion about which character (of several that look alike) is actually intended, may be associated with the presentation form of DNS names. Others may be linked to databases associated with the DNS, e.g., with the difficulty of finding an entry in a "Whois file" when it is not clear how to enter or to search for the characters that make up a name. This document discusses a family of restrictions on the names that can be registered. Restrictions of the type described can be imposed by a DNS zone ("registry"). The document also describes some possible tools for implementing such restrictions.

While the increased number and types of characters made available by Unicode considerably increases the scale of the potential problems, the problems addressed by this document are not new. No plausible set of restrictions will eliminate all problems and sources of confusion: for example, it has often been pointed out that, even in ASCII, the characters digit-one ("1") and lower case L ("l") can easily be confused in some display fonts. But, to the degree to which security may be aided by sensible risk reduction, these techniques may be helpful.

10. Acknowledgements

Discussions in the process of developing the JET Guidelines were vital in developing this document and all of the JET participants are consequently acknowledged. Attempts to explain some of the issues uncovered there to, and feedback from, Vint Cerf, Wendy Rickard, and members of the ICANN IDN Committee were also helpful in the thinking leading up to this document.

An effort by Paul Hoffman to create a generic specification for registration restrictions of this type helped to inspire this document, which takes a somewhat different, more language-oriented, approach than his initial draft. While the initial version of that draft indicated that multiple languages (or multiple language tables) for a single zone were infeasible, more recent versions [Hoffman-reg] shifted to inclusion of language-based approaches. The current version of this document incorporates considerable text, and even more ideas, from those drafts, with Paul Hoffman's generous permission.

Feedback was provided by several registry operators (of both country code and generic TLDs), including Edmon Chung and Ram Mohan of Afilias, and by ICANN and IANA staff, notably Tina Dam and Theresa Swinehart. This feedback about issues encountered in registering tables and designing IDN implementations resulted in the addition of significant clarifying text to the current version of the document.

The opinions expressed here are the sole responsibility of the author. Some of those whose ideas and comments are reflected in this document may disagree with the conclusions the author has drawn from them. The first draft version of this document was posted in June 2003.

11. Informative References

[Daniels] P.T. Daniels and W. Bright, The World's Writing Systems, Oxford: Oxford University Press: 1996.

[Drucker] Drucker, J., "The Alphabetic Labyrinth: The Letters in History and Imagination", 1995.

[Hoffman-reg] Hoffman, P., "A Method for Registering Internationalized Domain Names", Work in Progress, October 2003.

[IESG-IDN] Internet Engineering Steering Group, IETF, "IESG Statement on IDN", IESG Statement available from http://www.ietf.org/IESG/STATEMENTS/IDNstatement.txt, February 2003.

[ICANN-IDN] Internet Corporation for Assigned Names and Numbers (ICANN), "Guidelines for the Implementation of Internationalized Domain Names, Version 1.0", June 2003.

[ICANN-IDN2] Internet Corporation for Assigned Names and Numbers (ICANN), "Guidelines for the Implementation of Internationalized Domain Names, Version 2.0", September 2005.

[IANA-language-registry] Internet Assigned Numbers Authority (IANA), "IDN Language Table Registry", April 2004.

[LTRU-Registry] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", Work in Progress, October 2005.

[RFC952] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, October 1985.

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

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

[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.

[RFC3536] Hoffman, P., "Terminology Used in Internationalization in the IETF", RFC 3536, May 2003.

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

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

[Unicode32] The Unicode Consortium, "Unicode Standard Annex #28: Unicode 3.2", March 2002.

Author's Address

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

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

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.