INTERNET DRAFT   Editors: James SENG
draft-jseng-idn-admin-05.txt   John C KLENSIN, Wendy RICKARD
17 October 2003   Authors: K. KONISHI
Expires April 2004   K. HUANG, H. QIAN, Y. KO

Internationalized Domain Names Registration and Administration Guideline for Chinese, Japanese, and Korean

Status of This Memo

Abstract

Achieving internationalized access to domain names raises many complex issues. These are associated not only with basic protocol design--such as how names are represented on the network, compared, and converted to appropriate forms--but also with issues and options for deployment, transition, registration, and administration.

The IETF Standards for Internationalized Domain Names, known as "IDNA", focuses on access to domain names in a range of scripts that is broader in scope than the original ASCII. The development process made it clear that use of characters with similar appearances and/or interpretations created potential for confusion, as well as difficulties in deployment and transition. The conclusion was that, while those issues were important, they, could best be addressed administratively rather than through restrictions embedded in the protocols. This document defines a set of guidelines for applying restrictions of that type for CJK scripts and the zones that use them and, perhaps, the beginning of a framework for thinking about other zones, languages, and scripts.

Table of Contents

1. Introduction

Domain names form the fundamental naming architecture of the Internet. Countless Internet protocols and applications rely on them, not just for stability and continuity, but also to avoid ambiguity. They were designed to be identifiers without any language context. However, as domain names have become visible to end users through Web URLs and e-mail addresses, the strings in domain-name labels are being increasingly interpreted as names, words, or phrases. It is likely that users will do the same with languages of differing character sets--such as Chinese, Japanese and Korean (CJK)--in which many words or concepts are represented using short sequences of characters.

The introduction of what are called Internationalized Domain Names (IDN) amplifies both the difficulty of putting names into identifiers and the confusion that exists between scripts and languages. Character symbols that appear (or actually are) identical, or that have similar or identical semantics, but that are assigned the different code points, further increase the potential for confusion. DNS internationalization also affects a number of Internet protocols and applications and creates additional layers of complexity in terms of technical administration and services. Given the added complications of using a much broader range of characters than the original small ASCII subset, precautions are

necessary in the deployment of IDNs in order to minimize confusion and fraud.

The IETF IDN Working Group [IDN-WG] addressed the problem of handling the encoding and decoding of Unicode strings into and out of Domain Name System (DNS) labels with the goal that its solution would not put the operational DNS at any risk. Its work resulted in one primary protocol and three supporting ones, respectively:

1. Internationalizing Host Names in Applications [IDNA]
2. Preparation of Internationalized Strings [STRINGPREP]
3. A Stringprep Profile for Internationalized Domain Names [NAMEPREP]
4. Punycode [PUNYCODE]

IDNA--which calls on the others--normalizes and transforms strings that are intended to be used as IDNs. In combination, the four provide the minimum functions required for internationalization, such as performing case mappings, eliminating character differences that would cause severe problems, and specifying matching (equality). They also convert between the resulting Unicode code points and an ASCII-based form that is more suitable for storing in actual DNS labels. In this way, the IDNA transformations improve a user's chances of getting to the correct IDN.

Addressing the issues around differing character sets, a primary consideration and administrative challenge involves region-specific definitions, interpretations, and the semantics of strings to be used in IDNs. A Unicode string may have a specific meaning as a name, word, or phrase in a particular language but that meaning could vary depending on the country, region, culture, or other context in which the string is used. It might also have different interpretations in different languages that share some or all of the same characters. Therefore, individual zones and zone administrators may find it necessary to impose restrictions and procedures to reduce the likelihood of confusion--and instabilities of reference--within their own environments.

Over the centuries, the evolution of CJK characters--and the differences in their use in different languages and even in different regions where the same language is spoken--has given rise to the idea of "variants", wherein one conceptual character can be identified with several different Code Points in character sets for computer use. This document provides a framework for handling such variants while minimizing the possibility of serious user confusion in the obtaining or use of domain names. However, the concept of variants is complex and may require many different layers of solution, this guideline offers only one of the solution components. It is not sufficient by itself to solve the whole problem, even with zone-specific tables as described below.

Additionally, because of local language or writing-system differences, it is impossible to create universally accepted definitions for which potential variants are the same and which are not the same. It is even more difficult to define a technical algorithm to generate variants that are linguistically accurate--that is, that the variant forms produced make as much sense in the language as the originally specified forms. It is also possible that variants generated may have no meaning in the associated language or languages. The intention is not to generate meaningful "words" but to generate similar variants to be reserved. So even though the method described in this document may not always be

linguistically accurate--or need to be--it increases the chances of getting the right variants while accepting the inherent limitations of the DNS and the complexities of human language.

This document outlines a model for such conventions for zones in which labels that contain CJK characters are to be registered and a system for implementing that model. It provides a mechanism that allows each zone to define its own local rules for permitted characters and sequences and the handling of IDNs and their variants.

The document is an effort of the Joint Engineering Team (JET), a group composed of members of CNNIC, TWNIC, KRNIC, and JPNIC as well as other individual experts. It offers guidelines for zone administrators--including but not limited to registry operators and registrars?and information for all domain names holders on the administration of domain names that contain characters drawn from Chinese, Japanese, and Korean scripts. Other language groups are encouraged to develop their own guidelines as needed, based on these guidelines if that is helpful.

2. Definitions, Context, and Notation

2.1. Definitions and Context

This document uses a number of special terms. In this section, definitions and explanations are grouped topically. Some readers may prefer to skip over this material, returning, perhaps via the index to terminology in section 7, when needed.

2.1.1. IDN: The term "IDN" has a number of different uses: (a) as an

abbreviation for "Internationalized Domain Name"; (b) as a fully qualified domain name that contains at least one label that contains characters not appearing in ASCII, specifically not in the subset of ASCII recommended for domain names (the so-called "hostname" or "LDH" subset, see RFC1035 [STD13]); (c) as a label of a domain name that contains at least one character beyond ASCII; (d) as a Unicode string to be processed by Nameprep; (e) as a string that is an output from Nameprep; (f) as a string that is the result of processing through both Nameprep and conversion into Punycode; (g) as the abbreviation of an IDN (more properly, IDL) Package, in the terminology of this document; (h) as the abbreviation of the IETF IDN Working Group; (g) as the abbreviation of the ICANN IDN Committee; and (h) as standing for other IDN activities in other companies/organizations.

Because of the potential confusion, this document uses the term "IDN" as an abbreviation for Internationalized Domain Name and, specifically, in the second sense described in (b) above. It uses "IDL," defined immediately below, to refer to Internationalized Domain Labels.

2.1.2. **IDL: This document provides a guideline to be applied on a per-**zone basis, one label at a time. Therefore, the term "Internationalized Domain Label" or "IDL" will be used instead of the more general term "IDN" or its equivalents. The processing specifications of this document may be applied, in some zones, to ASCII characters also, if those characters are specified as valid in a Language Variant Table (see below). Hence, in some zones, an IDL may contain or consist entirely of "LDH" characters.

2.1.3. **FQDN: A fully qualified domain name, one that explicitly contains** all labels, including a Top-Level Domain (TLD) name. In this context, a TLD name is one whose label appears in a nameserver record in the root zone. The term "Domain Name Label" refers to any label of a FQDN.

2.1.4. **Registration: In this document, the term "registration" refers to** the process by which a potential domain name holder requests that a label be placed in the DNS either as an individual name within a domain or as a subdomain delegation from another domain name holder. In the case of a successful registration, the label or delegation records are placed in the relevant zone file, or, more specifically, they are "activated" or made "active" and additional IDLs may be reserved as part of an "IDL Package" (see below). The guidelines presented here are recommended for all zones--at any hierarchy level--in which CJK characters are to appear and not just domains at the first or second level.

2.1.5. **RFC3066: A system, widely used in the Internet, for coding and** representing names of languages. It is based on an International Organization for Standardization (ISO) standard for coding language names [ISO639], but expands it to provide additional precision.

2.1.6. **ISO/IEC 10646: The international standard universal multiple-**
octet coded character set ("UCS") [IS10646]. The Code Point definitions of this standard are identical to those of corresponding versions of the Unicode standard (see below). Consequently, the characters and their coding are often referred to as "Unicode characters."

2.1.7. **Unicode Character: The term "Unicode character" is used here in**
reference to characters chosen from the Unicode Standard Version 3.2 [UNICODE] (and hence from ISO/IEC 10646). In this document, the characters are identified by their positions, or "Code Points." The notation U+12AB, for example, indicates the character at the position 12AB (hexadecimal) in the Unicode 3.2 table. For characters in positions above FFFF, i.e., requiring more than sixteen bits to represent--a five to eight-character string is used, such as U+112AB for the character in position 12AB of plane 1.

2.1.8. **Unicode String: "Unicode string" refers to a string of Unicode**
characters. The Unicode string is identified by the sequence of the Unicode characters regardless of the encoding scheme.

2.1.9. **CJK Characters: CJK characters are characters commonly used in**
the Chinese, Japanese, or Korean languages, including but not limited to those defined in the Unicode Standard as ASCII (U+0020 to U+007F), Han ideographs (U+3400 to U+9FAF and U+20000 to U+2A6DF), Bopomofo (U+3100 to U+312F and U+31A0 to U+31BF), Kana (U+3040 to U+30FF), Jamo (U+1100 to 11FF and U+3130 to U+318F), Hangul (U+AC00 to U+D7AF and U+3130 to U+318F), and the respective compatibility forms. The particular characters that are permitted in a given zone are specified in the Language Variant Table(s) for that zone.

2.1.10. **Label String: A generic term referring to a string of characters**
that is a candidate for registration in the DNS or such a string, once registered. A label string may or may not be valid according to the rules of this specification and may even be invalid for IDNA use. The term "label", by itself, refers to a string that has been validated and may be formatted to appear in a DNS zone file.

2.1.11. **Language Variant Table: The key mechanisms of this specification**
utilize a three-column table, called a Language Variant Table, for each language permitted to be registered in the zone. Those columns are known, respectively, as "Valid Code Point", "Preferred Variant", and "Character Variant", which are defined separately below. The Language Variant Tables are critical to the success of the guideline described in this document. However, the principles to be used to generate the tables are not within the scope of this document and should be worked out by each registry separately (perhaps by adopting or adapting the work of some other registry). In this document, "Table" and "Variant Table" are used as short forms for Language Variant Table.

2.1.12. **Valid Code Point: In a Language Variant Table, the list of Code**

Points that is permitted for that language. Any other Code Points, or any string containing them, will be rejected by this specification. The Valid Code Point list appears as the first column of the Language Variant Table.

2.1.13. **Preferred Variant: In a Language Variant Table, a list of Code** Points corresponding to each Valid Code Point and providing possible substitutions for it. These substitutions are "preferred" in the sense that the variant labels generated using them are normally registered in the zone file, or "activated." The Preferred Code Points appear in column 2 of the Language Variant Table. "Preferred Code Point" is used interchangeably with this term.

2.1.14. **Character Variant: In a Language Variant Table, a second list of** Code Points corresponding to each Valid Code Point and providing possible substitutions for it. Unlike the Preferred Variants, substitutions based on Character Variants are normally reserved but not actually registered (or "activated"). Character Variants appear in column 3 of the Language Variant Table. The term "Code Point Variants" is used interchangeably with this term.

2.1.15. **Preferred Variant Label: A label generated by use of Preferred** Variants (or Preferred Code Points).

2.1.16. **Character Variant Label: A label generated by use of Character** Variants.

2.1.17. **Zone Variant: A Preferred or Character Variant Label that is** actually to be entered (registered) into the DNS--that is, into the zone file for the relevant zone. Zone Variants are also referred to as Zone Variant Labels or Active (or Activated) Labels.

2.1.18. **IDL Package: A collection of IDLs as determined by these** Guidelines. All labels in the package are "reserved", meaning they cannot be registered by anyone other than the holder of the Package. These reserved IDLs may be "activated", meaning they are actually entered into a zone file as a "Zone Variant". The IDL Package also contains identification of the language(s) associated with the registration process. The IDL and its variant labels form a single, atomic unit.

2.2 Notation for Ideographs and Other Non-ASCII CJK Characters.

For purposes of clarity, particularly in regard to examples, Han

ideographs appear in several places in this document. However, they do not appear in the ASCII version of this document. For the convenience of readers of the ASCII version--and some readers not familiar with recognizing and distinguishing Chinese characters--most uses of these characters will be associated with both their Unicode Code Points and an "asterisk tag" with its corresponding Chinese Romanization [ISO7098], with the tone mark represented by a number from 1 to 4. Those tags have no meaning outside this document; they are a quick visual and reading reference to help facilitate the combinations and transformations of characters in the guideline and table excerpts.

3. Scope of the Administrative Guidelines

Zone administrators are responsible for the administration of the domain name labels under their control. A zone administrator might be responsible for a large zone, such as a top-level domain (TLD)--whether generic or country code--or a smaller one, such as a typical second- or third-level domain. A large zone is often more complex than its smaller counterpart. However, actual technical administrative tasks--such as addition, deletion, delegation, and transfer of zones between domain name holders--are similar for all zones.

This document provides guidelines for the ways CJK characters should be handled within a zone, for how language issues should be considered and incorporated, and for how Domain Name Labels containing CJK characters should be administered (including registration, deletion, and transfer of labels).

Other IDN policies--such as the creation of new top-level domains (TLDs), the cost structure for registrations, and how the processes described here get allocated between registrar and registry if the zone makes that distinction--also are outside the scope of this document.

Technical implementation issues are not discussed here either. For example, deciding which guidelines should be implemented as registry actions and which should be registrar actions is left to zone administrators, with the possibility that it will differ from zone to zone.

3.1. Principles Underlying These Guidelines

In many places, in the event of a dispute over rights to a name (or, more accurately, DNS label string), this document assumes "first-come, first-served" (FCFS) as a resolution policy even though FCFS is not listed below as one of the principles for this document. If policies are already in place governing priorities and "rights", one can use the guidelines here by replacing uses of FCFS in this document with policies specific to the zone. Some of the guidelines here may not be applicable to other policies for determining rights to labels. Still other alternatives--such as use of UDRP [WIPO-UDRP] or mutual exclusion--might have little impact on other aspects of these guidelines.

(a) Although some Unicode strings may be pure identifiers made up of an assortment of characters from many languages and scripts, IDLs are likely to be "words" or "names" or "phrases" that have specific meaning in a language. While a zone administration might or might not require "meaning" as a registration criterion, meaning could prove to be a useful tool for avoiding user confusion.

Each IDL to be registered should be associated administratively with one or more languages.

Language associations should either be predetermined by the zone administrator and applied to the entire zone or be chosen by the registrants on a per-IDL basis. The latter may be necessary for some zones, but it will make administration more difficult and will increase the likelihood of conflicts in variant forms.

A given zone might have multiple languages associated with it or it may have no language specified at all. Omitting specification of a language may provide additional opportunities for user confusion and is therefore NOT recommended.

(b) Each language uses only a subset of Unicode characters. Therefore, if an IDL is associated with a language, it is not permitted to contain any Unicode character that is not within the valid subset for that language.

Each IDL to be registered must be verified against the valid subset of Unicode for the language(s) associated with the IDL. That subset is specified by the list of characters appearing in the first column of the language and zone-specific tables as described later in this document.

If the IDL fails this test for any of its associated languages, the IDL is not valid for registration.

Note that this verification is not necessarily linguistically accurate, because some languages have special rules. For example, some languages impose restrictions on the order in which particular combinations of characters may appear. Characters that are valid for the language--and hence permitted by this specification--might still not form valid words or even strings in the language.

(c) When an IDL is associated with a language, it may have Character Variants that depend on that language associated with it in addition to any Preferred Variants. These variants are potential sources of confusion with the Code Points in the original label string. Consequently, the labels generated from them should be unavailable to registrants of other names, words, or phrases.

During registration, all labels generated from the Character Variants for the associated language(s) of the IDL should be reserved.

IDL reservations of the type described here normally do not appear in the distributed DNS zone file. In other words, these reserved IDLs may not resolve. Domain name holders could request that these reserved IDLs be placed in the zone file and made active and resolvable.

Zones will need to establish local policies about how they are to be made active. Specifically, many zones, especially at the top level, have prohibited or restricted the use of "CNAME"s--DNS aliases--especially CNAMEs that point to nameserver delegation records (NS records). And long-term use of long-term aliases for domain hierarchies, rather than

single names ("DNAME records") are considered problematic because of the recursion they can introduce into DNS lookups.

(d) When an IDL is a "name", "word", or "phrase", it will have Character Variants depending on the associated language. Furthermore, one or more of those Character Variants will be used more often than others for linguistic, political, or other reasons. These more commonly used variants are distinguished from ordinary Character Variants and are known as Preferred Variant(s) for the particular language.

> To increase the likelihood of correct and predictable resolution of the IDN by end users, all labels generated from the Preferred Variants for the associated language(s) should be resolvable.

In other words, the Preferred Variant Labels should appear in the distributed DNS zone file.

(e) IDLs associated with one or more languages may have a large number of Character Variant Labels or Preferred Variant Labels. Some of these labels may include combinations of characters that are meaningless or invalid linguistically. It may therefore be appropriate for a zone to adopt procedures that include only linguistically-acceptable labels in the IDL Package.

> A zone administrator may impose additional rules and other processing activities to limit the number of Character Variant Labels or Preferred Variant Labels that are actually reserved or registered.

These additional rules and other processing activities are based on policies and/or procedures imposed on a per-zone basis and therefore are not within the scope of this document. Such policies or procedures might be used, for example, to restrict the number of Preferred Variant Labels actually reserved or to prevent certain words from being registered at all.

(f) There are some Character Variant Labels and Preferred Variant Labels that are associated with each IDL. These labels are considered "equivalent" to each another. To avoid confusion, they all should be assigned to a single domain name holder.

> The IDL and its variant labels should be grouped together into a single atomic unit, known in this document as an "IDL Package".

The IDL Package is created upon registration and is atomic: Transfer and deletion of an IDL is performed on the IDL Package as a whole. That is, an IDL within the IDL Package may not be transferred or deleted individually; any re-registration, transfers, or other actions that impact the IDL should also affect the other variants.

The name-conflict resolution policy associated with this zone could result in a conflict with the principle of IDL Package atomicity. In such a case, the policy must be defined to make the precedence clear.

3.2. Registration of IDL

To conform to the principles described in 3.1, this document introduces two concepts: the Language Variant Table and the IDL Package. These are

described in the next two subsections, followed by a description of the algorithm that is used to interpret the table and generate variant labels.

3.2.1. Using the Language Variant Table

For each zone that uses a given language, each language should have its own Language Variant Table. The table consists of a header section that identifies references and version information, followed by a section with one row for each Code Point that is valid for the language and three columns.

(1) The first column contains the subset of Unicode characters that is valid to be registered ("Valid Code Point"). This is used to verify the IDL to be registered (see 3.1b). As in the registration procedure described later, this column is used as an index to examine characters that appear in a proposed IDL to be processed. The collection of Valid Code Points in the table for a particular language can be thought of as defining the script for that language, although the normal definition of a script would not include, for example, ASCII characters with CJK ones.

(2) The second column contains the Preferred Variant(s) of the corresponding Unicode character in column one ("Valid Code Point"). These variant characters are used to generate the Preferred Variant Labels for the IDL. Those labels should be resolvable (see 3.1d). Under normal circumstances, all of those Preferred Variant Labels will be activated in the relevant zone file so that they will resolve when the DNS is queried for them.

(3) The third column contains the Character Variant(s) for the corresponding Valid Code Point. These are used to generate the Character Variant Labels of the IDL, which are then to be reserved (see 3.1c). Registration--or activation--of labels generated from Character Variants will normally be a registrant decision, subject to local policy.

Each entry in a column consists of one or more Code Points, expressed as a numeric character number in the Unicode table and optionally followed by a parenthetical reference. The first column--or Valid Code Point--may have only one Code Point specified in a given row. The other columns may have more than one.

Any row may be terminated with an optional comment, starting in "#".

The formal syntax of the table and more-precise definitions of some of its organization appear in Section 5.

The Language Variant Table should be provided by a relevant group, organization, or body. However, the question of who is relevant or has the authority to create this table and the rules that define it is beyond the scope of this document.

3.2.2. IDL Package

The IDL Package is created on successful registration and consists of:

(1) the IDL registered

(2) the language(s) associated with the IDL

(3) the version of the associated character variant table

(4) the reserved IDLs

(5) active IDLs--that is, "Zone Variant Labels" that are to appear in the DNS zone file

3.2.3. Procedure for Registering IDLs

An explanation follows each step.

```
Step 1.    IN <= IDL to be registered and
           {L} <= Set of languages associated with IN
```

Start the process with the label string (prospective IDL) to be registered and the associated language(s) as input.

```
Step 2.    Generate the Nameprep-processed version of the IN, applying
           all mappings and canonicalization required by IDNA.
```

The prospective IDL is processed by using Nameprep to apply the normalizations and exclusions globally required to use IDNA. If the Nameprep processing fails, then the IDL is invalid and the registration process must stop.

```
Step 2.1.  NP(IN) <= Nameprep processed IN
Step 2.2.  Check availability of NP(IN).
```

```
          If not available, route to conflict policy.
```

The Nameprep-processed IDL is then checked against the contents of the zone file and previously created IDL Packages. If it is already registered or reserved, then a conflict exists that must be resolved by applying whatever policy is applicable for the zone. For example, if FCFS is used, the registration process terminates unless the conflict resolution policy provides another alternative.

```
Step 3. Process each language.
        For each language (AL} in {L}
```

Step 3 goes through all languages associated with the proposed IDL and checks each character (after Nameprep has been applied) for validity in each of them. It then applies the Preferred Variants (column 2 values) and the Character Variants (column 3 values) to generate candidate labels.

```
Step 3.1. Check validity of NP(IN) in AL. If failed, stop processing.
```

In step 3.1, IDL validation is done by checking that every Code Point in the Nameprep-processed IDL is a Code Point allowed by the "Valid Code Point" column of the Character Variant Table for the language. This is then repeated for any other languages (and hence, Language Variant Tables) specified in the registration. If one or more Code Points are not valid, the registration process terminates.

```
Step 3.2. PV(IN,AL) <= Set of available Nameprep-processed Preferred
                       Variants of NP(IN) in AL
```

Step 3.2 generates the list of Preferred Variant Labels of the IDL by doing a combination (see Step 3.2A below) of all possible variants listed in the "Preferred Variant(s)" column for each Code Point in the Nameprep-processed IDL. The generated Preferred Variant Labels must be processed through Nameprep. If the Nameprep processing fails for any Preferred Variant Label (this is unlikely to occur if the Preferred Variants [Code Points] are processed through Nameprep before being placed in the table), then that variant label will be removed from the list. The remaining Preferred Variant Labels in the list are then checked to see whether they are already registered or reserved. If any are registered or reserved, then the conflict resolution policy will apply. In general, this will not prevent the originally requested IDL from being registered unless the policy prevents such registration. For example, if FCFS is applied, then the conflicting variants will be removed from the list, but the originally requested IDL and any remaining variants will be registered (see steps 5 and 8 below).

Step 3.2A Generating variant labels from Variant Code Points.

Steps 3.2 and 3.3 require that the Preferred Variants and Character Variants be combined with the original IDL to form sets of variant labels. Conceptually, one starts with the original, Nameprep-processed, IDL and examines each of its characters in turn. If a character is encountered for which there is a corresponding Preferred Variant or Character Variant, a new variant label is produced with the Variant Code Point substituted for the original one. If variant labels already exist as the result of the processing of characters that appeared earlier in the original IDL, then the substitutions are made in them as well, resulting in additional generated variant labels. This operation is repeated separately for the Preferred Variants (in Step 3.2) and Character Variants (in Step 3.3). Of course, equivalent results could be achieved by processing the original IDL's characters in order, building the Preferred Variant Label set and Character Variant Label set in parallel.

This process will sometimes generate a very large number of labels. For example, if only two of the characters in the original IDL are associated with Preferred Variants and if the first of those characters has three Preferred Variants and the second has two, one ends up with 12 variant labels to be placed in the IDL Package and, normally, in the zone file. Repeating the process for Character Variants, if any exist, would further increase the number of labels. And if more than one language is specified for the original IDL, then repetition of the process for additional languages (see step 4, below) might further increase the size of the set.

For illustrative purposes, the "combination" process could be achieved by a recursive function similar to the following pseudocode:

```
Function Combination(Str)
  F <= first codepoint of Str
  SStr <= Substring of Str, without the first code point
  NSC <= {}

  If SStr is empty then
   for each V in (Variants of code point F)
     NSC = NSC set-union (the string with the code point V)
   End of Loop
  Else
    SubCom = Combination(SStr)
    For each V in (Variants of code point F)
      For each SC in SubCom
        NSC = NSC set-union (the string with the
            first code point V followed by the string SC)
      End of Loop
    End of Loop
  Endif

  Return NSC
```

Step 3.3. CV(IN,AL) <= Set of available Nameprep-processed Character

Variants of NP(IN) in AL

This step generates the list of Character Variant Labels by doing a combination (see Step 3.2A above) of all the possible variants listed in the "Character Variant(s)" column for each Code Point in the Nameprep-processed original IDL. As with the Preferred Variant Labels, the generated Character Variant Labels must be processed by, and acceptable to, Nameprep. If the Nameprep processing fails for a Character Variant Label, then that variant label will be removed from the list. The remaining Character Variant Labels are then checked to be sure they are not registered or reserved. If one or more are, then the conflict resolution policy is applied. As with Preferred Variant Labels, a conflict that is resolved in favor of the earlier registrant does not, in general, prevent the IDL from being registered, nor the remaining variants from being reserved in step 6 below.

Step 3.4. End of Loop

Step 4. Let PVall be the set-union of all PV(IN,AL)

Step 4 generates the Preferred Variants Label for all languages.
In this step, and again in step 6 below, the zone administrator may impose additional rules and processing activities to restrict the number of Preferred (tentatively to be reserved and activated) and Character (tentatively to be reserved) Label Variants. These additional rules and processing activities are zone policy specific and therefore are not specified in this document.

Step 5. {ZV} <= PVall set-union NP(IN)

Step 5 generates the initial Zone Variants. The set includes all Preferred Variants for all languages and the original Nameprep-processed IDL. Unless excluded by further processing, these Zone Variants will be activated--that is, placed into the DNS zone. Note that the "set-union" operation will eliminate any duplicates.

Step 6. Let CVall be the set-union of all CV(IN,AL), set-minus {ZV}

Step 6 generates the Reserved Label Variants (the Character Variant Label set). These labels are normally reserved but not activated. The set includes all Character Variant Labels for all languages, but not the Zone Variants defined in the previous step. The set-union and set-minus operations eliminate any duplicates.

Step 7. Create IDL Package for IN using IN, {L}, {ZV} and CVall

In Step 7, the "IDL Package" is created using the original IDL, the associated language(s), the Zone Variant Labels, and the Reserved Variant Labels. If zone-specific additional processing or filtering is to be applied to eliminate linguistically inappropriate or other forms, it should be applied before the IDL Package is actually assembled.

Step 8. Put {ZV} into zone file

The activated IDLs are converted via ToASCII with UseSTD13ASCIIRules [IDNA] before being placed into the zone file. This conversion results in the IDLs being in the actual IDNA ("Punycode") form used in zone files, while the IDLs have been carried in Unicode form up to this point. If ToASCII fails for any of the activated IDLs, that IDL must not be placed into the zone file. If the IDL is a subdomain name, it will be delegated.

3.3. Deletion and Transfer of IDL and IDL Package

In traditional domain administration, every Domain Name Label is independent of all other Domain Name Labels. Registration, deletion, and transfer of labels is done on a per-label basis. However, with the guidelines discussed here, each IDL is associated with specific languages, with all label variants--both active (zone) and reserved--together in an IDL Package. This quite deliberately prohibits labels that contain sufficient mixtures of characters from different scripts to make them impossible as words in any given language. If a zone chooses to not impose that restriction--that is, to permit labels to be constructed by picking characters from several different languages and scripts--then the guidelines described here would be inappropriate.

As stated earlier, the IDL package should be treated as a single atomic unit and all variants of the IDL should belong to a single domain-name holder. If the local policy related to the handling of disagreements requires a particular IDL to be transferred and deleted independently of the IDL Package, the conflict policy would take precedence. In such an event, the conflict policy should include a transfer or delete procedure that takes the nature of IDL Packages into consideration.

When an IDL Package is deleted, all of the Zone and Reserved Label Variants again become available. The deletion of one IDL Package does not change any other IDL Packages.

3.4. Activation and Deactivation of IDL variants

Because there are active (registered) IDLs and inactive (reserved but not registered) IDLs within an IDL package, processes are required to activate or deactivate IDL variants within an IDL Package.

3.4.1. Activation Algorithm

Step 1. IN <= IDL to be activated and PA <= IDL Package

Start with the IDL to be activated and the IDL Package of which it is a member.

Step 2. NP(IN) <= Nameprep processed IN

Process the IDL through Nameprep. This step should never cause a problem, or even a change, since all labels that become part of the IDL Package are processed through Nameprep in Step 3.2 or 3.3 of the Registration procedure (section 3.2.3).

Step 3. If NP(IN) not in {RV} then stop

Verify that the Nameprep-processed version of the IDL appears as a still-unactivated label in the IDL Package, i.e., in the list of Reserved Label Variants, {RV}. It might be a useful "sanity check" to also verify that it does not already appear in the zone file.

Step 4. {RV} <= {RV} set-minus NP(IN) and {ZV} <= {ZV} set-union NP(IN)

Within the IDL Package, remove the Nameprep-processed version of the IDL from the list of Reserved Label Variants and add it to the list of active (zone) label variants.

Step 5. Put {ZV} into the zone file

Actually register (activate) the Zone Variant Labels.

3.4.2. Deactivation Algorithm

Step 1. IN <= IDL to be deactivated and PA <= IDL Package

As with activation, start with the IDL to be deactivated and the IDL Package of which it is a member.

Step 2. NP(IN) <= Nameprep processed IN

Get the Nameprep-processed version of the name (see discussion in the previous section).

Step 3. If NP(IN) not in {ZV} then stop

Verify that the Nameprep-processed version of the IDL appears as an activated (zone) label variant in the IDL Package. It might be a useful "sanity check" at this point to also verify that it actually appears in the zone file.

Step 4. {RV} <= {RV} set-union NP(IN) and {ZV} <= {ZV} set-minus NP(IN)

Within the IDL Package, remove the Nameprep-processed version of the IDL from the list of Active (Zone) Label Variants and add it to the list of Reserved (but inactive) Label Variants.

Step 5. Put {ZV} into the zone file

3.5. Managing Changes in Language Associations

Since the IDL package is an atomic unit and the associated list of variants must not be changed after creation, this document does not include a mechanism for adding and deleting language associations within the IDL package. Instead, it recommends deleting the IDL package entirely, followed by a registration with the new set of languages. Zone administrators may find it desirable to devise procedures that prevent other parties from capturing the labels in the IDL Package during these operations.

3.6. Managing Changes to the Language Variant Tables

Language Variant Tables are subject to changes over time, and these changes may or may not be backward compatible. It is possible that updated Language Variant Tables may produce a different set of Preferred Variants and Reserved Variants.

In order to preserve the atomicity of the IDL Package, when the Language Variant Table is changed, IDL Packages created using the previous version of the Language Variant Table must not be updated or affected.

4. Examples of Guideline Use in Zones

To provide a meaningful example, some Language Variant Tables must be defined. Assume, then, for the purpose of giving examples, that the following four Language Variant Tables are defined:

Note: these tables are not a representation of the actual tables, and they do not contain sufficient entries to be used in any actual implementation.

a) Language Variant Table for zh-cn and zh-sg

```
Reference 1 CP936 (commonly known as GBK)
Reference 2 zVariant, zTradVariant, zSimpVariant in Unihan.txt
Reference 3 List of Simplified character Table (Simplified column)
Reference 4 zSimpVariant in Unihan.txt
Reference 5 variant that exists in GB2312, common simplified hanzi

Version 1 20020701 # July 2002

56E2(1);56E2(5);5718(2)           # sphere, ball, circle; mass, lump
5718(1);56E2(4);56E2(2),56E3(2)   # sphere, ball, circle; mass, lump
60F3(1);60F3(5);                  # think, speculate, plan, consider
654E(1);6559(5);6559(2)           # teach
6559(1);6559(5);654E(2)           # teach, class
6DF8(1);6E05(5);6E05(2)           # clear
6E05(1);6E05(5);6DF8(2)           # clear, pure, clean; peaceful
771E(1);771F(5);771F(2)           # real, actual, true, genuine
771F(1);771F(5);771E(2)           # real, actual, true, genuine
8054(1);8054(3);806F(2)           # connect, join; associate, ally
806F(1);8054(3);8054(2),8068(2)   # connect, join; associate, ally
```

```
96C6(1);96C6(5);                  # assemble, collect together
```

b) Language Variant Table for zh-tw

```
Reference 1 CP950 (commonly known as BIG5)
Reference 2 zVariant, zTradVariant, zSimpVariant in Unihan.txt
Reference 3 List of Simplified Character Table (Traditional column)
Reference 4 zTradVariant in Unihan.txt

Version 1 20020701 # July 2002

5718(1);5718(4);56E2(2),56E3(2)   # sphere, ball, circle; mass, lump
60F3(1);60F3(1);                  # think, speculate, plan, consider
6559(1);6559(1);654E(2)           # teach, class
6E05(1);6E05(1);6DF8(2)           # clear, pure, clean; peaceful
771F(1);771F(1);771E(2)           # real, actual, true, genuine
806F(1);806F(3);8054(2),8068(2)   # connect, join; associate, ally
96C6(1);96C6(1);                  # assemble, collect together
```

c) Language Variant Table for ja

```
Reference 1 CP932 (commonly known as Shift-JIS)
Reference 2 zVariant in Unihan.txt
Reference 3 variant that exists in JIS X0208, commonly used Kanji

Version 1 20020701 # July 2002

5718(1);5718(3);56E3(2)           # sphere, ball, circle; mass, lump
60F3(1);60F3(3);                  # think, speculate, plan, consider
654E(1);6559(3);6559(2)           # teach
6559(1);6559(3);654E(2)           # teach, class
6DF8(1);6E05(3);6E05(2)           # clear
6E05(1);6E05(3);6DF8(2)           # clear, pure, clean; peaceful
771E(1);771E(1);771F(2)           # real, actual, true, genuine
771F(1);771F(1);771E(2)           # real, actual, true, genuine
806F(1);806F(1);8068(2)           # connect, join; associate, ally
96C6(1);96C6(3);                  # assemble, collect together
```

d) Language Variant Table for ko

```
Reference 1 CP949 (commonly known as EUC-KR)
Reference 2 zVariant and K-source in Unihan.txt

Version 1 20020701 # July 2002

5718(1);5718(1);56E3(2)           # sphere, ball, circle; mass, lump
60F3(1);60F3(1);                  # think, speculate, plan, consider
654E(1);654E(1);6559(2)           # teach
6DF8(1);6DF8(1);6E05(2)           # clear
771E(1);771E(1);771F(2)           # real, actual, true, genuine
806F(1);806F(1);8068(2)           # connect, join; associate, ally
```

```
96C6(1);96C6(1);                  # assemble, collect together

Example 1: IDL = (U+6E05 U+771F U+6559) *qing2 zhen1 jiao4*
           {L} = {zh-cn, zh-sg, zh-tw}

NP(IN) = (U+6E05 U+771F U+6559)
PV(IN,zh-cn) = (U+6E05 U+771F U+6559)
PV(IN,zh-sg) = (U+6E05 U+771F U+6559)
PV(IN,zh-tw) = (U+6E05 U+771F U+6559)

{ZV} = {(U+6E05 U+771F U+6559)}
CVall = {(U+6E05 U+771E U+6559),
        (U+6E05 U+771E U+654E),
        (U+6E05 U+771F U+654E),
        (U+6DF8 U+771E U+6559),
        (U+6DF8 U+771E U+654E),
        (U+6DF8 U+771F U+6559),
        (U+6DF8 U+771F U+654E)}

Example 2: IDL = (U+6E05 U+771F U+6559) *qing2 zhen1 jiao4*
           {L} = {ja}

NP(IN) = (U+6E05 U+771F U+6559)
PV(IN,ja) = (U+6E05 U+771F U+6559)
{ZV} = {(U+6E05 U+771F U+6559)}

CVall = {(U+6E05 U+771E U+6559),
        (U+6E05 U+771E U+654E),
        (U+6E05 U+771F U+654E),
        (U+6DF8 U+771E U+6559),
        (U+6DF8 U+771E U+654E),
        (U+6DF8 U+771F U+6559),
        (U+6DF8 U+771F U+654E)}

Example 3: IDL = (U+6E05 U+771F U+6559) *qing2 zhen1 jiao4*
           {L} = {zh-cn, zh-sg, zh-tw, ja, ko}

NP(IN) = (U+6E05 U+771F U+6559) *qing2 zhen1 jiao4*
Invalid registration because U+6E05 is invalid in L = ko

Example 4: IDL = (U+806F U+60F3 U+96C6 U+5718)
                 *lian2 xiang3 ji2 tuan2*
          {L} = {zh-cn, zh-sg, zh-tw}

NP(IN) = (U+806F U+60F3 U+96C6 U+5718)
PV(IN,zh-cn) = (U+8054 U+60F3 U+96C6 U+56E2)
PV(IN,zh-sg) = (U+8054 U+60F3 U+96C6 U+56E2)
PV(IN,zh-tw) = (U+806F U+60F3 U+96C6 U+5718)
{ZV} = {(U+8054 U+60F3 U+96C6 U+56E2),
       (U+806F U+60F3 U+96C6 U+5718)}
CVall = {(U+8054 U+60F3 U+96C6 U+56E3),
        (U+8054 U+60F3 U+96C6 U+5718),
        (U+806F U+60F3 U+96C6 U+56E2),
        (U+806f U+60F3 U+96C6 U+56E3),
        (U+8068 U+60F3 U+96C6 U+56E2),
        (U+8068 U+60F3 U+96C6 U+56E3),
        (U+8068 U+60F3 U+96C6 U+5718)

Example 5: IDL = (U+8054 U+60F3 U+96C6 U+56E2)
               *lian2 xiang3 ji2 tuan2*
          {L} = {zh-cn, zh-sg}

NP(IN) = (U+8054 U+60F3 U+96C6 U+56E2)
PV(IN,zh-cn) = (U+8054 U+60F3 U+96C6 U+56E2)
PV(IN,zh-sg) = (U+8054 U+60F3 U+96C6 U+56E2)
{ZV} = {(U+8054 U+60F3 U+96C6 U+56E2)}
CVall = {(U+8054 U+60F3 U+96C6 U+56E3),
        (U+8054 U+60F3 U+96C6 U+5718),

        (U+806F U+60F3 U+96C6 U+56E2),
        (U+806f U+60F3 U+96C6 U+56E3),
        (U+806F U+60F3 U+96C6 U+5718),
        (U+8068 U+60F3 U+96C6 U+56E2),
        (U+8068 U+60F3 U+96C6 U+56E3),
        (U+8068 U+60F3 U+96C6 U+5718)}

Example 6: IDL = (U+8054 U+60F3 U+96C6 U+56E2)
               *lian2 xiang3 ji2 tuan2*
           {L} = {zh-cn, zh-sg, zh-tw}

NP(IN) = (U+8054 U+60F3 U+96C6 U+56E2)
Invalid registration because U+8054 is invalid in L = zh-tw

Example 7: IDL = (U+806F U+60F3 U+96C6 U+5718)
               *lian2 xiang3 ji2 tuan2*
           {L} = {ja,ko}

NP(IN) = (U+806F U+60F3 U+96C6 U+5718)
PV(IN,ja) = (U+806F U+60F3 U+96C6 U+5718)
PV(IN,ko) = (U+806F U+60F3 U+96C6 U+5718)
{ZV} = {(U+806F U+60F3 U+96C6 U+5718)}
CVall = {(U+806F U+60F3 U+96C6 U+56E3),
        (U+8068 U+60F3 U+96C6 U+5718),
        (U+8068 U+60F3 U+96C6 U+56E3)}
```

5. Syntax Description for the Language Variant Table

The formal syntax for the Language Variant Table is as follows, using the IETF "ABNF" metalanguage [ABNF]. Some comments on this syntax appear immediately after it.

5.1 ABNF Syntax

```
LanguageVariantTable = 1*ReferenceLine VersionLine 1*EntryLine
```

```
ReferenceLine = "Reference" SP RefNo SP RefDesciption [ Comment ] CRLF
RefNo = 1*DIGIT
RefDesciption = *[VCHAR]
VersionLine = "Version" SP VersionNo SP VersionDate [ Comment ] CRLF
VersionNo = 1*DIGIT
VersionDate = YYYYMMDD
EntryLine = VariantEntry/Comment CRLF

VariantEntry = ValidCodePoint  ";"
               PreferredVariant ";" CharacterVariant [ Comment ]
ValidCodePoint = CodePoint
RefList = RefNo  0*( "," RefNo )
PreferredVariant = CodePointSet 0*( "," CodePointSet )
CharacterVariant = CodePointSet 0*( "," CodePointSet )
CodePointSet = CodePoint 0*( SP CodePoint )
CodePoint = 4*8DIGIT  [ "(" Reflist ")" ]
Comment = "#" *VCHAR
```

YYYYMMDD is an integer, in alphabetic form, representing a date, where YYYY is the 4-digit year, MM is the 2-digit month, and DD is the 2-digit day.

5.2. Comments and Explanation of Syntax

Any lines starting with, or portions of lines after, the hash symbol("#") are treated as comments. Comments have no significance in the processing of the tables; nor are there any syntax requirements

between the hash symbol and the end of the line. Blank lines in the tables are ignored completely.

Every language should have its own Language Variant Table provided by a relevant group, organization, or other body. That table will normally be based on some established standard or standards. The group that defines a Language Variant Table should document references to the appropriate standards at the beginning of the table, tagged with the word "Reference" followed by an integer (the reference number) followed by the description of the reference. For example:

```
Reference 1 CP936 (commonly known as GBK)
Reference 2 zVariant, zTradVariant, zSimpVariant in Unihan.txt
Reference 3 List of Simplified Character Table (Simplified column)
Reference 4 zSimpVariant in Unihan.txt
Reference 5 Variant that exists in GB2312, common simplified Hanzi
```

Each Language Variant Table must have a version number and its release date. This is tagged with the word "Version" followed by an integer then followed by the date in the format YYYYMMDD, where YYYY is the 4-digit year, MM is the 2-digit month, and DD is the 2-digit day of the publication date of the table.

```
Version 1 20020701     # July 2002 Version 1
```

The table has three columns, separated by semicolons: "Valid Code Point"; "Preferred Variant(s)"; and "Character Variant(s)".

The "Valid Code Point" is the subset of Unicode characters that are valid to be registered.

There can be more than one Preferred Variant; hence there could be multiple entries in the "Preferred Variant(s)" column. If the "Preferred Variant(s)" column is empty, then there is no corresponding Preferred Variant; in other words, the Preferred Variant is null. Unless local policy dictates otherwise, the procedures above will result in only those labels that reflect the valid code point being activated (registered) into the zone file.

The "Character Variant(s)" column contains all Character Variants of the Code Point. Since the Code Point is always a variant of itself, to avoid redundancy, the Code Point is assumed to be part of the "Character Variant(s)" and need not be repeated in the "Character Variant(s)" column.

If the variant in the "Preferred Variant(s)" or the "Character Variant(s)" column is composed of a sequence of Code Points, then sequence of Code Points is listed separated by a space.

If there are multiple variants in the "Preferred Variant(s)" or the "Character Variant(s)" column, then each variant is separated by a comma.

Any Code Point listed in the "Preferred Variant(s)" column must be allowed by the rules for the relevant language to be registered. However, this is not a requirement for the entries in the "Character Variant(s)" column; it is possible that some of those entries may not be allowed to be registered.

Every Code Point in the table should have a corresponding reference number (associated with the references) specified to justify the entry. The reference number is placed in parentheses after the Code Point. If there is more than one reference, then the numbers are placed within a single set of parentheses and separated by commas.

6. Security Considerations

As discussed in the Introduction, substantially-unrestricted use of international (non-ASCII) characters in domain name labels may cause user confusion and invite various types of attacks. In particular, in the case of CJK languages, an attacker has an opportunity to divert or confuse users as a result of different characters (or, more specifically, assigned code points) with identical or similar semantics. These Guidelines provide a partial remedy for those risks by supplying a framework for prohibiting inappropriate characters from being registered at all and for permitting "variant" characters to be grouped together and reserved, so that they can only be registered in the DNS by the same owner. However, the system it suggests is no better or worse than the per-zone and per-language tables whose format and use this document specifies. Specific tables, and any additional local processing, will reflect per-zone decisions about the balance between risk and flexibility of registrations.   And, of course, errors in construction of those tables may significantly reduce the quality of protection provided.

7. Index to Terminology

As a convenience to the reader, this section lists all of the special terminology used in this document, with a pointer to the section in which it is defined.

8. Acknowledgments

The authors gratefully acknowledge the contributions of:

- V. CHEN, N. HSU, H. HOTTA, S. TASHIRO, Y. YONEYA, and other Joint Engineering Team members at the JET meeting in Bangkok, Thailand.

- Yves Arrouye, an observer at the JET meeting in Bangkok, for his contribution on the IDL Package.

- Those who commented on, and made suggestions about, earlier versions, including Harald ALVESTRAND, Erin CHEN, Patrik FALTSTROM, Paul HOFFMAN, Soobok LEE, LEE Xiaodong, MAO Wei, Erik NORDMARK, and L.M. TSENG.

9. Authors' Addresses

James SENG
180 Lompang Road
#22-07 Singapore 670180
Phone: +65 9638-7085
E-mail: jseng@pobox.org.sg

Kazunori KONISHI
JPNIC
Kokusai-Kougyou-Kanda Bldg 6F
2-3-4 Uchi-Kanda, Chiyoda-ku
Tokyo 101-0047
Japan
Phone: +81 49-278-7313
E-mail: konishi@jp.apan.net

Kenny HUANG
TWNIC
3F, 16, Kang Hwa Street, Taipei
Taiwan
TEL : 886-2-2658-6510
E-mail: huangk@alum.sinica.edu

QIAN Hualin
CNNIC
No.6 Branch-box of No.349 Mailbox, Beijing 100080
Peoples Republic of China
E-mail: Hlqian@cnnic.net.cn

KO YangWoo
PeaceNet
Yangchun P.O. Box 81 Seoul 158-600
Korea
E-mail: yw@mrko.pe.kr

John C KLENSIN
1770 Massachusetts Avenue, No. 322
Cambridge, MA 02140
U.S.A.
E-mail: Klensin+ietf@jck.com

Wendy RICKARD
The Rickard Group
16 Seminary Ave
Hopewell, NJ 08525
USA
E-mail: rickard@rickardgroup.com

10. Normative References

[ABNF]   Crocker, D. and P. Overell, eds.,Augmented BNF for Syntax Specifications: ABNF, RFC 2234 November 1997.

[STD13]   Mockapetris, P. "Domain names--concepts and facilities" (RFC 1034) and "Domain names--implementation and specification" (RFC 1035), STD 13, November 1987.

[RFC3066]   Alvestrand, H., Tags for the Identification of Languages, RFC3066, Jan 2001.

[IDNA]   Faltstrom, Patrik, Paul Hoffman, Adam M. Costello, Internationalizing Domain Names in Applications (IDNA), RFC 3490, March 2003.

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

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

[NAMEPREP] Hoffman, P. and M. Blanchet, Nameprep: A Stringprep Profile for Internationalized Domain Names, RFC 3491, March 2003.

[IS10646]   A product of ISO/IEC JTC1/SC2/WG2, Work Item JTC1.02.18 (ISO/IEC 10646). It is a multipart standard: Part 1,

published as ISO/IEC 10646-1:2000(E), covers the Architecture and Basic Multilingual Plane, and Part 2, published as ISO/IEC 10646-2:2001(E), covers the supplementary (additional) planes.

[UNIHAN]   Unicode Han Database, Unicode Consortium ftp://ftp.unicode.org/Public/UNIDATA/Unihan.txt.

[UNICODE] The Unicode Consortium, "The Unicode Standard--Version 3.0," ISBN 0-201-61633-5. Unicode Standard Annex #28 (http://www.unicode.org/unicode/reports/tr28/) defines Version 3.2 of the Unicode Standard, which is definitive for IDNA and this document.

[ISO7098] ISO 7098;1991 Information and documentation--Romanization of Chinese, ISO/TC46/SC2.

11. Nonnormative References

[IDN-WG] IETF Internationalized Domain Names Working Group, now concluded,idn@ops.ietf.org, James Seng, Marc Blanchet, co-chairs, http://www.i-d-n.net/.

[IESG-IDN] Internet Engineering Steering Group, IETF, "IESG Statement on IDN", 11 February 2003, http://www.ietf.org/IESG/STATEMENTS/IDNstatement.txt.

[ISO639] "ISO 639:1988 (E/F)--Code for the representation of names of languages"--International Organization for Standardization, 1st edition, 1988-04-01.