INTERNET DRAFT
draft-jseng-idn-admin-01.txt
18th Oct 2002
Expires 18th April 2003

Editors: James SENG
John KLENSIN
Authors: K. KONISHI
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 include not only associated with basic protocol design (i.e., how the names are represented on the network, compared, and converted to appropriate forms) but also issues and options for deployment, transition, registration and administration.

The IETF IDN working group focused on the development of a standards track specification for access to domain names in a broader range of scripts than the original ASCII. It became clear during its efforts that there was great potential for confusion, and difficulties in deployment and transition, due to characters with similar appearances or interpretations and that those issues could best be addressed administratively, rather than through restrictions embedded in the protocols.

This document provides guidelines for zone administrators (including but not limited to registry operators and registrars), and information for all domain names holders, on the administration of those domain names which contain characters drawn from Chinese, Japanese and Korean scripts (CJK). Other language groups are encouraged to develop their own guidelines as needed, based on these guideline if that is helpful.

Comments on this document can be sent to the authors at idn-admin@jdna.jp.

Table of Contents

0. Pre-Note for ASCII-version of this document

In order to make meanings clear, especially in examples, Han ideographs are used in several places in this document. Of course, these ideographs do not appear in its ASCII form of this document. So, for the convenience of readers of the ASCII format and some readers not familiar with recognizing and distinguishing Chinese characters, each use of a particular character will be associated with both its Unicode code point and an "asterisk tag" with its corresponding Chinese Romanization [ISO7098] with the tone mark represented by a number 1 to 4. Those tags have no meaning outside this document; they are intended simply to provide a quick visual and reading reference to facilitate the combinations and transformations of characters in the guideline and table excerpts. Appendix A would provide the Romanization of the ideographs in Japanese (ISO 3602) and Korean (ISO 11941).

1. Introduction

Defining and specifying protocols for Internationalized Domain Names has been one of the most controversial tasks initiated by the IETF in recent years. Domain names are the fundamental naming architecture of the Internet; many Internet protocols and applications rely on the stability, continuity, and absence of ambiguity of the DNS.

The introduction of internationalized domain names (IDN) amplifies the difficulty of putting names into identifiers and the confusion between scripts and languages. It impacts many internet protocols and applications and creates more complexity in technical administration and services.

While the IETF IDN working group [IDN-WG] focused on the technical problems of IDN, administrative guidelines are also important in order to reduce unnecessary user confusion and domain name disputes among domain name holders.

The IDN working group has completed working group last call for the following internet-drafts:

1. Preparation of Internationalized Strings [STRINGPREP]
2. Internationalizing Host Names In Applications [IDNA]
3. Punycode version 0.3.3 [PUNYCODE]
4. A Stringprep Profile for Internationalized Domain Names [NAMEPREP]

These drafts specify that the intersystem protocols that make up the domain name system infrastructure remain unchanged. Instead, they introduce internationalization (I18N) [Note1] in client software (particularly via the IDNA protocol) using an ASCII Compatible Encoding (ACE) known as Punycode.

The domain name protocols [STD13] also specify that characters are to be interpreted so that upper and lower case Latin-based characters are considered equivalent. But with the introduction of Unicode characters beyond US-ASCII, and the possibility to represent a single character in multiple ways in ISO10646/Unicode [UNICODE], a normalization process, known as Nameprep, has been proposed to handle the more complex problems of character-matching for those additional characters. Nameprep is also executed by client software as described in IDNA.

While Nameprep normalizes domain names so that the users have an improved chance of getting the right domain name from information provided in other forms, as required for I18N, Nameprep does not handle any localization (L10N).

This becomes significant when a domain name holder attempts to use a Unicode string forming a "name", "word", or "phrase" that may have certain meaning in a certain language or when used as a domain name. Such Unicode string may have different variants in the context of the language or culture.

Generally, these localized variants in CJK can be classified into four categories, as described by Halpern et al. [C2C]: [Note2]

a. Character (or Code) variants

Character (or Code) variants refer to variants that are generated by character-by-character (or code-by-code) substitution.

An example in English would be "A" or "a" (U+0041 or U+0061).
Two examples in Chinese would be 飛 U+98DB *fei1* or 飞 U+98DE *fei1* and 機 U+6A5F *ji1* or 机 U+673A *ji1*.

Note that this does not mean the choice between U+6A5F and U+673A is always symmetric like the one between "A" and "a" – it is a choice only for Chinese but not for Japanese.

The variants for particular characters may be just to drop them. For example, points and vowels characters in Hebrew (U+05B0 to U+05C4) and Arabic (U+064B to U+0652) are optional; the variants for strings containing them are constructed by simply dropping those points and vowels.

Code variants may also occur when different code points are assigned to what visually or abstractly are the "same" character, possibility due to compatibility issues, type face differences or script range. For example, LATIN CAPITAL LETTER A (U+0041) normally has an appearance identical to GREEK CAPTIAL LETTER A (U+0391). CJK scripts have font variants for compatibility (either U+4E0D or U+F967 may be used) and "zVariant" (e.g. U+5154 and U+514E).

The difficulty lies in defining which characters are the "same" and which are not.

b. Orthographic variants

Orthographic variants refer to variants that are generated by word-by-word substitution.

An example in English would be "color" and "colour".

It is possible for some of these orthographic variants to be generated by character variants. For example "airplane" in Chinese may be either 飛機 U+98DB U+6A5F *fei1 ji1* or 飞机 U+98DE U+673A *fei1 ji1*.

Other orthographic variants may not be generated by character variants. For example, in Chinese, both "發" U+767C *fa1* and "髮" U+9AEE *fa4* are related to "发" U+53D1 *fa1 or fa4* depending on the word. For hair, "头发" U+5934 U+53D1 *tou2 fa4*, the variant should be "頭髮" U+982D U+9AEE *tou2 fa4* but not "頭發" U+982D U+767C *tou2 fa1*.

c. Lexemic variants

Lexemic variants refer to variants that can be generated when language is considered, by word-by-word substitution.

An example in English would be cab, taxi, or taxicab.

An example in Chinese would be 資訊 U+8CC7 U+8A0A *zi1 xun4* or 信息 U+4FE1 U+606F *xin4 xi1*.

Note that there is no relationship between U+8CC7 and U+4FE1 or U+8A0A and U+606F, i.e., the sequence 資息 U+8CC7 U+606F *zi1 xi1* does not exist in Chinese.

d. Contextual variants

Contextual variants refer to variants that are generated by word-by-word substitutions with context considered.

In English, the word "plane" has different meanings and could be replaced by with different equivalent words (synonyms) such as "airplane" or "plane" (as in a flat-surface or device for smoothing wood) depending on context. And, of course, "plain", which is pronounced the same way, and indistinguishable in speech-to-text contexts such as computer input systems for the visually impaired, is a different word entirely.

Similarly, the word 文件 U+6587 U+4EF6 *wen2 jian4* could be either document 文件 U+6587 U+4EF6 *wen2 jian4*or data file 檔案 U+6A94 U+6848 *dang3 an4* depending on context.

Although domain names were designed to be identifiers without any language context, users have not been prevented from using strings in domain names and interpreting them as "words" or "names". It is likely that users will do this with IDN as well. Therefore, given the added complications of using a much broader range of characters, precautions will be required when deploying IDN to minimize confusion and fraud.

The intention of these guidelines is to provide advice about the deployment of IDNs, with language consideration, but focusing only on the category of character variants to increase the possibility of successful resolution and reduced confusion while accepting inherent DNS limitations.

2. Definitions

Unless otherwise stated, the definitions of the terms used in this document are consistent with "Terminology Used in Internationalization in the IETF" [I18NTERMS].

"FQDN" refers to a fully-qualified domain name and "domain name label" refers to a label of a FQDN.

RFC3066 [RFC3066] defines a system for coding and representing languages.

ISO/IEC 10646 is a universal multiple-octet coded character set that is a product of ISO/IEC JTC1/SC2/WG2, Work Item JTC1.02.18 (ISO/IEC 10646). It is a multi-part standard: Part 1, published as ISO/IEC 10646-1:2000(E) covering the Architecture and Basic Multilingual Plane; Part 2, published as ISO/IEC 10646-2:2001(E) covers the supplementary (additional) planes.

The Unicode Consortium publishes "The Unicode Standard – Version 3.0", ISBN 0-201-61633-5. In March 2002, Unicode Consortium published Unicode

Standard Annex #28. That annex defines Version 3.2 of The Unicode Standard, which is fully synchronized with ISO/IEC 10646-1:2000 (with Amendment 1).

The term "Unicode character" is used here to refer to characters chosen from The Unicode Standard Version 3.2 (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.

Similarly, "Unicode string" refers to a string of Unicode characters. The Unicode string is identify by the sequence of the Unicode characters regardless of the encoding scheme.

The term "IDN" is often used to refer to many different things: (a) an abbreviation for "Internationalized Domain Name" (b) a fully-qualified domain name that contains at least one label that contains characters not appearing in ASCII (c) a label of a domain name that contains at least one character beyond ASCII (d) a Unicode string to be processed by Nameprep (e) an IDN Package (in this document context) (f) a Nameprep processed string (g) a Nameprep and Punycode processed string (h) the IETF IDN Working Group (g) ICANN IDN Committee (h) other IDN activities in other companies/organizations etc.

Because of the potential confusion, this document shall use the term "IDN" as an abbreviation for "Internationalized Domain Name" only.

And also, this document provides a guideline to be applied on a per zone basis, one label at a time, the term "Internationalized Domain Name Label" or "IDL" will be used instead.

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 sub-domain delegation from another domain name holder. A successful registration would then lead to the label or delegation records being placed in the relevant zone file. The guidelines presented here are recommended for all zones, at any hierarchy level, in which CJK characters are to appear, not just domains at the first or second level.

CJK characters are characters commonly used in Chinese, Japanese or Korean language including but not limited to ASCII (U+0020 to U+007F, Han Ideograph (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 its respective compatibility forms.

3. Administrative Framework

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), generic or country code, or a smaller one such as a typical second or third level domain. A large zone would often be more complex then a smaller one (sometimes it is just larger). However, normally, actual technical administrative tasks -- such as addition, deletion, delegation and

transfer of zones between domain name holders -- are similar for all zones.

At the same time, different zones may have different policies and processes. For example, a pay-per-domain policy and registry/registrar model for .COM may not be applicable to such domains as .SG or .IBM.COM. The latter, for example, has very restricted policies about who is permitted to have a domain name label under IBM.COM, the types of string that are permitted, and different procedures for obtaining those string.

This document only provides guidelines for how CJK characters should be handled within a zone, how language issues should be considered and incorporated, and how domain name labels containing CJK characters should be administered (including registration, deletion and transfer of labels). It does not provide any guidance for handling of non-CKJ characters or languages in zones.

Other IDN policies, as the creation of new TLDs, or the cost structure for registrations, are outside the scope of this document. Such discussions should be conducted in forums outside the IETF as well.

Technical implementation issues are not discussed here either. For example, the decision as to whether various of the guidelines should be implemented as registry or registrar actions is left to zone administrators, possibly differing from zone to zone.

3.1. Principles underlying these Guidelines

In many places, this document would assumes "First-Come-First-Serve" (FCFS) as a conflict policy in the event of a dispute although FCFS is not listed as one of the principles. If other policies dominate priorities and "rights", one can use these guidelines by replacing uses of FCFS in this document by appropriate other policy rules specific to the zone. In other cases, some of these guidelines may not be applicable although, some alternatives for determining rights to labels -- such as use of UDRP or mutual exclusion -- might have little impact on other aspects of these guidelines.

(a) Each IDL to be registered should be associated with one or more languages.

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 names or phrases that have certain meaning in some language. While a zone administration might or might not require "meaning" as a registration criterion, the possibility of meaning provides a useful tool when trying to avoid user confusion.

Zone administrators should administratively associate one or more language with each IDL. These associations should either be pre-determined by the zone administrator and applied to the entire zone or chosen by the registrants on a per-IDL basis. The latter may be necessary for some zones, but 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 have no language specified at all, but doing so may provide additional opportunities for user confusion, and is therefore not recommended.

The zone administrator must also verify the validity of the IDL requested by using information associated with the chosen language and possibly other rules as appropriate.

(b) When an IDL is registered, all of the character variants for the associated language(s) should be reserved for the registrant. Each language associated with the IDL will lead to different character variants.

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

Since different languages may imply different sets of variants, the IDLs reserved for one IDL may overlap those reserved for another. In this case, the reserved IDLs should be bound to one registration or the other, or excluded from both, according to the applicable registration or dispute resolution policy for the zone.

(c) For a given base language, the IDL may have one or more recommended variants that should be suggested to the domain name holder for active registration as synonyms.

Some language rules may prefer certain variants over others. To increase the likelihood of correct and predictable resolution of the IDL by end-users, the recommended variants should be active.

(d) The IDL and its reserved variants with the language(s) association must be atomic.

The IDL and its reserved variants for the associated language(s) are to be considered as a single unit – an "IDL Package". For a given IDL, that IDL package is defined by these guidelines and created upon registration.

The IDL Package is atomic: Transfer and deletion of IDL are performed on the IDL Package as a whole. IDL, either active or reserved, within the IDL Package must not be transferred or deleted individually. I.e., any re-registration, transfers, or other actions that impact the IDL should also impact the reserved variants. Separate registration or other actions for the variants are not possible if these guidelines are to accomplish their purpose.

Conflict policy of the zone may result in violation of the IDL Package atomicity. In such case, the conflict policy would take precedence.

3.2. Registration of IDL

Conforming to the principles described in 3.1, the registration of an IDL would require at least two components, i.e., the character variant tables for the language and the registration algorithm.

3.2.1. Language character variant table

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 a character variant table provided by a relevant group (or organization or other body) and based on established standards. The group that defines a particular character variant table should document references to the appropriate standards in beginning of 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 character variant table must have a version number. 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 fields, separated by semicolons. The fields are: "valid code point"; "recommended variant(s)"; and "character variant(s)".

Only code points listed in the "valid code point" field are allowed to be registered as part of a IDL associated with that language.

There can be one or more "recommended variant(s)" (i.e., entries in the "recommended variant(s)" column). If the "recommended variant(s)" column is empty, then there is no corresponding variant.

The "character variant(s)" column contains all variants of the code point, including but not limited to the code point itself and the "recommended variant(s)".

If the variant is composed of a sequence of code points, then sequence of code points is listed separated by a space in the "recommended variant(s)" or "character variant(s)".

If there are multiple variants, each variant must be separated by a comma in the "recommended variant(s)" or "character variant(s)".

Any code point listed in the "recommended 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.

3.2.2. Formal syntax

This section uses the IETF "ABNF" metalanguage [ABNF]

```
LanguageCharacterVariantTable = 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 [ "(" RefList ") ] ;" RecommendedVariant
";" CharacterVariant [ Comment ]
ValidCodePoint = CodePoint
RefList = RefNo  0*( "," RefNo )
RecommendedVariant = CodePointSet 0*( "," CodePointSet )
CharacterVariant = CodePointSet 0*( "," CodePointSet )
CodePointSet = CodePoint 0* ( SP CodePoint )
CodePoint = 4DIGIT [DIGIT] [DIGIT]
Comment = "#" *VCHAR
```

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

3.2.3. Registration Algorithm

(An explanation of these steps follows them)

```
1.    IN <= IDL to be registered and
      {L} <= Set of languages associated with IN
2.    {V} <= Set of version numbers of the language character
             variant tables derived from {L}
3.    NP(IN) <= Nameprep processed IN  and
      check availability of NP(IN).
      If not available, route to conflict policy.
4.    For each AL in {L}
4.1.    Check validity of NP(IN) in AL. If failed, stop processing.
4.2.    PV(IN,AL) <= Set of available Nameprep processed recommended
                     variants of NP(IN) in AL
4.3.    RV(IN,AL) <= Set of available Nameprep processed character
                     variants of NP(IN) in AL
4.4.  End of Loop
5.    {PV} <= Set of all PV(IN,AL) with optional processing.
6.    {ZV} <= {PV} set-union NP(IN)
7.    {RV} <= Set of all RV(IN,AL) set-minus {ZV}
8.    Create IDL Package for IN using IN, {L}, {V}, {ZV} and {RV}
9.    Put {ZV} into zone file
```

Explanation

Step 1 takes the IDL to be registered and the associated language(s) as input to the process.

Step 2 extract the set of version numbers of the associated language(s) tables.

Step 3 Nameprep processed the IDL. If the Nameprep processed IDL is already registered or reserved, then the conflict policy is applied here. For example, if FCFS is used, the registration process would stop here.

Step 4 goes through all languages associated with the proposed IDL, checks for validity in each language, and generates the recommended variants and the reserved variants.

In step 4.1, 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. If one or more code points are invalid, the registration process must stop here.

Step 4.2 generates the list of recommended variants of the IDL by doing a combination of all possible variants listed in "recommend variant(s)" column for each code point in the Nameprep processed IDL. Generated variants must be processed with Nameprep. If any of the recommended variants of the IDL is registered or reserved, then the conflict policy will be applied although this does not prevent the IDL from being registered. For example, if FCFS is used, then the conflicting variant(s) will be removed from the list.

Step 4.3 generates the list of reserved variants by doing a combination of all the possible variants listed in "character variant(s)" column for each code point in the Nameprep processed IDL. Generated variants must be Nameprep processed. If any of the variants are registered or reserved, then the conflict policy will apply here although this does not prevent the IDL from being registered. For example, if FCFS is used, then the conflict variants will be removed from the list.

The "combination" in Step 4.2 and Step 4.3 could achieve by a recursive function similar to the following pseudo code:

```
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 5 generates the list of all recommended variants for all language. Optionally, the algorithm may reduce the list of recommended variants by prompting the user to select the recommended variants.

Step 6 generates the list of variants including the Nameprep processed IDL which to be activated and Step 7 generates the list of reserved variants.

Then an "IDL Package" for IDL is created in Step 8 with the original IDL, the associated language(s), all the list of activated IDLs and the list of variants. The version numbers of the language character variants tables are also stored in the IDL Package.

Lastly, the activated IDLs are converted using ToASCII [IDNA] with UseSTD13ASCIIRules on and then put into the zone file. If the IDL is a subdomain name, it will be delegated. The activated IDLs may be delegated to a different domain name server so long it is owned by the same domain name holder.

3.3. Deletion and Transfer of IDL and IDL Package

In normal domain administration, every domain name label is independent of all other domain name labels. Registration, deletion and transfer of domain name labels is done on a per domain name label basis. Depending on the zone's administrative policies, aliases (e.g., "CNAME" entries) may be bound to particular labels with rules about whether one can be changed without the other. Current policies in gTLDs generally prohibit registration of such aliases, in part to avoid needing to form and enforce policies about these change (or binding) rules.

However, with internationalization, each IDL is bound to a list of variant IDLs (with the list depending on the associated language), bound together in an IDL Package.

Because all variants of the IDL should belong to a single domain name holder, the IDL Package should be treated as a single entity. Individual IDL, either active or reserved, within the IDL Package must not be deleted or transferred independently of the other IDLs. Specifically, if an IDL is to be deleted or transferred, that action must be taken only as part of an action that affects the entire IDL Package.

If the local conflict policy requires IDL to be transferred and deleted independently of the IDL Package, the conflict policy would take precedence. In such event, the conflict policy should be associated with a transfer or delete procedure taking IDL Package into consideration.

When an IDL Package is deleted, all the active and reserved variants would be available again. IDL Package deletion does not change any other IDL Packages, including IDL Packages that have variants that conflict with the variants in the deleted IDL Package. This is to be consistent with the atomicity and predictability of the IDL Package.

3.4. Activation and De-activation of IDL variants

As there are active IDLs and inactive IDLs within an IDL Package, processes are required to activate or de-activate IDL variants in an IDL Package.

The activation algorithm is described below:

1. IN <= IDL to be activated & PA <= IDL Package
2. NP(IN) <= Nameprep processed IN
3. If NP(IN) not in {RV} then stop
4. {RV} <= {RV} set-minus NP(IN) and {ZV} <= {ZV} set-union NP(IN)
5. Put {ZV} into the zone file

Similarly, the deactivation algorithm:

1. IN <= IDL to be deactivated & PA <= IDL Package
2. NP(IN) <= Nameprep processed IN
3. If NP(IN) not in {ZV} then stop
4. {RV} <= {RV} set-union NP(IN) and {ZV} <= {ZV} set-minus NP(IN)
5. Put {ZV} into the zone file

3.5. Adding/Deleting language(s) association

The list of variants is generated from the IDL and tables for the associated languages. If the language associations are changed, then the lists of variants have to be updated. On the other hand, the IDL Package is atomic and the list of variants must not be changed after creation.

Therefore, this document recommends deleting the IDL Package followed by a registration with the new set of languages rather than attempting to add or delete language(s) association within the IDL Package. Zone administrators may find it desirable to devise procedures to prevent other parties from capturing the labels in the IDL Package during these operations.

3.6. Versioning of the language character variant tables

Language character variants tables are subjected to changes over time and the changes may or may not be backward compatible. It is possible that different version of the language character variants tables may produce a different set of recommended variants and reserved variants.

New IDL Packages should use the latest version of the language character variants tables.

Existing IDL Packages created using previous version of language character variants tables are not affected when there a new version of the character variants table is released.

4. Example of Guideline Adoption

To provide a meaningful example, some language character variant tables have to be defined. Assume, then, that the following four language character variants tables are defined (note that these tables are not a representation of the actual table and they do not contain sufficient entries to be used in any actual implementation):

a) language character variants tables 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 variants 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 variants 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 variants table for ko

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

Version 1 20020701 # July 2002

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

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

NP(IN) = 清真教 (U+6E05 U+771F U+6559)
PV(IN,zh-cn) = 清真教 (U+6E05 U+771F U+6559)
PV(IN,zh-sg) = 清真教 (U+6E05 U+771F U+6559)
PV(IN,zh-tw) = 清真教 (U+6E05 U+771F U+6559)
{ZV} = {清真教 (U+6E05 U+771F U+6559)}
{RV} = {清眞教 (U+6E05 U+771E U+6559),
        清眞教 (U+6E05 U+771E U+654E),
        清真教 (U+6E05 U+771F U+654E),
        淸眞教 (U+6DF8 U+771E U+6559),
        淸眞教 (U+6DF8 U+771E U+654E),
        淸真教 (U+6DF8 U+771F U+6559),
        淸真教 (U+6DF8 U+771F U+654E)}

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

NP(IN) = 清真教 (U+6E05 U+771F U+6559)
PV(IN,ja) = 清真教 (U+6E05 U+771F U+6559)
{ZV} = {清真教 (U+6E05 U+771F U+6559)}
{RV} = {清眞教 (U+6E05 U+771E U+6559),
        清眞教 (U+6E05 U+771E U+654E),
        清真教 (U+6E05 U+771F U+654E),
        淸眞教 (U+6DF8 U+771E U+6559),
        淸眞教 (U+6DF8 U+771E U+654E),
        淸真教 (U+6DF8 U+771F U+6559),
```

清真教 (U+6DF8 U+771F U+654E)}

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

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

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

NP(IN) = 聯想集團 (U+806F U+60F3 U+96C6 U+5718)
PV(IN,zh-cn) = 联想集团 (U+8054 U+60F3 U+96C6 U+56E2)
PV(IN,zh-sg) = 联想集团 (U+8054 U+60F3 U+96C6 U+56E2)
PV(IN,zh-tw) = 聯想集團 (U+806F U+60F3 U+96C6 U+5718)
{ZV} = {联想集团 (U+8054 U+60F3 U+96C6 U+56E2),
聯想集團 (U+806F U+60F3 U+96C6 U+5718)}
{RV} = {联想集团 (U+8054 U+60F3 U+96C6 U+56E3),
联想集團 (U+8054 U+60F3 U+96C6 U+5718),
聯想集团 (U+806F U+60F3 U+96C6 U+56E2),
聯想集团 (U+806f U+60F3 U+96C6 U+56E3),
聨想集团 (U+8068 U+60F3 U+96C6 U+56E2),
聨想集团 (U+8068 U+60F3 U+96C6 U+56E3),
聨想集團 (U+8068 U+60F3 U+96C6 U+5718)

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

NP(IN) = 联想集团 (U+8054 U+60F3 U+96C6 U+56E2)
PV(IN,zh-cn) = 联想集团 (U+8054 U+60F3 U+96C6 U+56E2)
PV(IN,zh-sg) = 联想集团 (U+8054 U+60F3 U+96C6 U+56E2)
{ZV} = {联想集团 (U+8054 U+60F3 U+96C6 U+56E2)}
{RV} = {联想集团 (U+8054 U+60F3 U+96C6 U+56E3),
联想集團 (U+8054 U+60F3 U+96C6 U+5718),
聯想集团 (U+806F U+60F3 U+96C6 U+56E2),
聯想集团 (U+806f U+60F3 U+96C6 U+56E3),
聯想集團 (U+806F U+60F3 U+96C6 U+5718),
聨想集团 (U+8068 U+60F3 U+96C6 U+56E2),
聨想集团 (U+8068 U+60F3 U+96C6 U+56E3),
聨想集團 (U+8068 U+60F3 U+96C6 U+5718)}

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

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

Example 7: IDL = 聯想集團 (U+806F U+60F3 U+96C6 U+5718)
lian2 xiang3 ji2 tuan2

{L} = {ja,ko}

NP(IN) = 聯想集團 (U+806F U+60F3 U+96C6 U+5718)
PV(IN,ja) = 聯想集團 (U+806F U+60F3 U+96C6 U+5718)
PV(IN,ko) = 聯想集團 (U+806F U+60F3 U+96C6 U+5718)
{ZV} = {聯想集團 (U+806F U+60F3 U+96C6 U+5718)}
{RV} = {聯想集团 (U+806F U+60F3 U+96C6 U+56E3),
聯想集團 (U+8068 U+60F3 U+96C6 U+5718),
聯想集团 (U+8068 U+60F3 U+96C6 U+56E3)}

i. Notes

1 The terms "i18n" and "l10n", sometimes used in upper-case form (i.e., "I18N" and "L10N"), have become popular in international standards usage as abbreviations for "internationalization" and "localization", respectively. The abbreviations were derived by using the first and last letters of the words, with the number of characters that appear between them. I.e., in "internationalization", there are 18 characters between the initial "i" and the terminal "n".

2. Every human language is unique and therefore, every linguistic and localization issue is also unique. It is difficult or impossible to make comparisons across multiple languages or to classify them into categories. And any cross-language analogies are, by their very nature, imperfect at best.

For example, to classify Traditional Chinese/Simplified Chinese as upper/lower case makes as much sense as to classify TC/SC as "spelling variant" like "color" and "colour". Both comparisons are potentially useful but neither is completely correct.

3. The variants in CJK are very complex and require many different layers of solution. This guideline is a one of the solution components, but not sufficient, by itself, to solve the whole problem.

ii. Acknowledgements

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.

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

Soobok LEE
L.M TSENG
Patrik FALTSTROM
Paul HOFFMAN
Erin CHEN
LEE Xiaodong
Harald ALVESTRAND

iii. Author(s)

James SENG

PSB Certification
3 Science Park Drive
#03-12 PSB Annex
Singapore 118233
Phone: +65 6885-1657
Email: 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
Email: konishi@jp.apan.net

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

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

KO YangWoo
PeaceNet
Yangchun P.O. Box 81 Seoul 158-600
Korea
Email: newcat@peacenet.or.kr

John C KLENSIN
1770 Massachusetts Ave, No. 322
Cambridge, MA 02140
USA
Email: Klensin+ietf@jck.com

iv. Appendix A

[How to read the Han Ideograph provided in this document. -- Will complete this section in next revision]

v. Normative References

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

[I18NTERMS] Terminology Used in Internationalization in the IETF, draft-hoffman-i18n-terms-07.txt, September 2002, Paul Hoffman, work in progress

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

[IDNA] Internationalizing Domain Names in Applications, draft-ietf-idn-idna, Feb 2002, Patrik Faltstrom, Paul Hoffman, Adam M. Costella, work in progress

[PUNYCODE] Punycode: An encoding of Unicode for use with IDNA, draft-ietf-idn-punycode, Feb 2002, Adam M. Costello, work in progress

[STRINGPREP] Preparation of Internationalized Strings, draft-hoffman-stringprep, Feb 2002, Paul Hoffman, Marc Blanchet, work in progress

[NAMEPREP] Nameprep: A Stringprep Profile for Internationalized Domain Names, work in progress, draft-ietf-idn-nameprep, Feb 2002, Paul Hoffman, Marc Blanchet, work in progress

[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.

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

vi. Non-normative References

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

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

[C2C] Pitfalls and Complexities of Chinese to Chinese Conversion, http://www.cjk.org/cjk/c2c/c2c.pdf, Jack Halpern, Jouni Kerman

vii. Other Issues

It is possible that many 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.

The language Character Variants tables are critical to the success of the guideline. A badly designed table may either generate too many meaningless variants or may not generate enough meaningful variants. The principles to be used to generate the tables are not within the scope of this document, nor are the tables themselves.

This document recommends against registration of IDL in a particular language until the language character variants table for that language is available.

Outstanding Issues

(1) Erin suggested (if I (JcK) correctly understood her) that, if multiple languages are associated with a given name, the recommended variant list for a given code point be treated as the intersection of the variant lists for each of the languages, not the union. As I understand the current algorithm, it effectively takes the union. Taking the intersection has the technical advantage that it would significantly reduce the number of variant strings that must be reserved. It also has the policy advantage of discouraging people from registering with multiple languages if they don't need to – otherwise, we will have everyone trying to register in all of the possibly-relevant languages, which would make this effort a good deal less effective than it might be.

Taking the intersection is also consistent with a rule that appears to exist now. As shown in Example 3, if an attempt is made to register a name and associate it with multiple languages, it must be valid in all of those languages or the registration attempt will fail. So we intersect the validity criteria on a language basis, and should probably intersect the variants.

But that is an algorithm change, since we have to extract the variant lists for each code point for each language, take the intersection, and then process against that, rather than against each language in turn.

[JS – I disagree in taking the intersection of the set. No doubt by doing intersection we will reduce the abuse of specifying multiple language to increase the set of reserved variants, our goal is precisely to reserve as much variants as possible for the domain name holder, not vice versa.

Suppose we have a string ABC with variants ABD ACD ABF in Chinese, ABE ACD in Japanese and CBD ACD in Korean.

Assuming a registrant register ABC in CJK, right now he will get the reserved set of {ABC, ACD, ABF, ABE, CBD}.

On the other hand, if we do intersection, this set will be reduced to {ACD}, leaving other variants like ABF, ABE and CBD open for potential conflict. And the only way he can protect this confusion is to individually register ABF, ABE and CBD manually individually, something we trying to prevent.]

[Further explanation by Erin:

I'm sorry maybe my previous suggestion is not clear enough.

I mean if multiple languages are associated with a given nanme, the range of valid code point sould be the intersection of all the associated languages.

But, if multiple languages are associated with a given nanme, the recommended variants should be take the union and put into zone file.

The same, the character variant code also sould be take the union for each of the languages.]

(2) A note went by indicating that the plan was to drop the Han characters from the IETF-submission version of this document. We can post I-Ds in PDF and publish RFCs in PDF and/or Postscript, as long as we provide ASCII. I find having the Han characters very useful, and trust that those of you who can read them find them even more so. So I would suggest that we hand off the pair of an ASCII document (with the Han characters removed) and a PDF document (that looks like the Word text we have been looking it) to the I-D editor. I've got full Acrobat here and can presumably produce the thing if needed.

(3) We still need to sort out the issue of whether reserving a variant that may (in a current or future table) conflict with another character, with the possibility of activating it is an invitation to cybersquatting and other abuses. That isn't clear, let me try an illustration: suppose we have a character X, with variants A, B, and C, and a character Y, with variants D and C. Now, if Y is registered first, then its package includes {Y*, D, C}, using the symbol "*" to denote an active name. When X is registered, its package consists of {X, A, B}. X's owner can't reserve or activate C, since it was reserved to Y. But much of the reason for doing all of this work was the concern that C can be confused with either Y or X. So doesn't this create an opportunity for Y to threaten, or extort money from, X by threatening to activate C?

[JS -- The conflict of X & Y over C in this case could be resolved by existing conflict policy. The revised guideline now makes it possible to modify the IDL Package in the event of dispute]

That problem gets worse, I think, if Erin's suggestion in (1) is not adopted. And I continue to believe that the only solution that will work is to prevent anyone from activating C. Or, more generally, at any given time, there will be a set of language variant tables that will be considered valid by the administrator of a particular zone. The zone administrator would take the union of all of those tables, using the "valid code point" as the key as usual, and then permanently reserve any character that appeared most than once in a variant column. Small matter of programming.

(4) In page 9, on the paragraph starting with "The character variant(s) column contains ..."

Page: 21
This seems to be saying that the code points listed in the third column will always be a proper superset of the union of the first and second columns. If that is correct, it violates a fundamental principle that I was taught about good programming and systems design – minimization of duplication of information, since such duplicates are error-prone. And, if I have not interpreted the intent correctly, the text needs to be fixed. Somehow.

[JS -- correct, it is duplicated. The duplication is bad from system design view but it makes it 'complete' and easy to explain.]