IDN Code Exchange Mapping Structure

Status of this memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

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

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

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

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

Abstract

The client side of IDN [IDN] has to accomdate users of different scripts, with many existing national and international standards and different clients and local servers. The server side of IDN is a proven stable US-ASCII only DNS system. An international Unicode standard based, national standard tabulation exchange structure called IDN-map is described.

Contents

1. Introduction
2. IDN Standards Code Exchange Table
   2.1 Structure of IDN Code Exchange Table
   2.2 Access of IDN Code Exchange Table
3. Version control and Language tags of IDN Code Exchange Table
   3.1 Language Tags
   3.2 Language Tag File Format
   3.3 Identification of a Tag of an Input String
4. Interface with IDN Code Exchange Map
   4.1 Language Specific Modules
   4.2 Script Specific Canonicalization
   4.3 Language Specific Normalization and Presentation
   4.4 Language Tagged IDN Label Conversions
   4.5 Uniform Idn-label Protocol
5. Prefered Embodiments of IDN Code Exchange Map

1. Introduction

Users from international travelers, to middle school students on Tibet Plateau, to librarians in Washington D.C. wish to have direct access to internet from their familiar desktop with their native languages for years, the internet community has been trying to bring that services to the users from many locations around the world. Some servers have successfully demonstrated the concept for such a service, for example, http://www.3721.com is providing Pinyin [Pinyin] based
mnemonic registration for Chinese users and allow clicking through on users' screen from Chinese URL[URL] window. This document suggests a client side structure and cooperate by servers to support such a direct and speedy universal URL access for all users on Internet.

1.1 Context

Symbols of natural languages are open sets for CJK[CJK] as well as for English [ALPHBET]. For example, Chinese continuously discovers characters, "Zi", to add onto their character set exceeding the number of 100,000 already. In the United States, many European symbols appears in American names, which makes its symbol set exceeds the original of English 26 letters. Combinations of symbols are called "word" in English, "ci" in Chinese, and "string" in term of domain names. In this document, the discussion is focused on a mapping structure, called IDN-map for symbols, which are referred as UCS[UCS] "Code Points", to specify its relationships among various national symbol standards in term of code points to support accurate, speedy combinations of symbols for Internet domain name identification.

Due to the nature of UCS character set as a multi-script, for multi-language users, besides the issue of equal speedy access, IDN-map has to address three additional issues in recognizing the nature of an open symbol set. The first issue is allowing more mixed script use when there is enough experience in dealing with existing mixed script use. The second issue is allowing new symbols to be added into the table in the future. The third issue is to let depreciated local standards drop out without implicating the international structure and IDN-map's life expectancy.

IDN-map needs two key mechanism to accommodate above issues in addition to current [nameprep] proposal. The first key mechanism is a traffic signal, called "Language Tag" [RFC 3066], since the users are using different spoken languages as they are defined in [ISO 639]. These languages are expressed with symbols specified in UCS[ISO 10646], as well as ASCII[ASCII], GB[GB], BIG5[BIG5], JIS[JIS], KSC[KSC], ISCII[ISCII]. The users dictate which symbol to be used and from where in the UCS, which exhibits very high locality for legitimate uses, and here is called "Script Range" of a specific language tag. A script range may include more than one code blocks of UCS, such that it permits the deployment of IDN in multiple stages, and allows a script range to be expanded in the future for mixed script use.

The second key of IDN-map is a two-level symbol switching mechanism, called language tagged ASCII compatible character encoding, short for T-ACE. The T, for the tag part, is the switch between different spoken languages which may implies various national and international standards including ASCII. The ACE part is the switch among symbols within the same script range. The ACE part of the switch is a massive one for Chinese tag: it is a range from 2,000 for student readers of "People’s Daily", to 50,000 and above for a librarian and many other variants in between, not including Japanese, Korean and other spoken languages. To provide a switching system for such a variation use of symbols, each switch in the system needs to be labeled for a human. It needs to be a mnemonic switch and it needs to be scaleble for different user groups too. The proposed ACE is a mnemonic encoding scheme, and is called StepCode [StepCode]. With T-ACE in a multiple standard tabulation, a simple uniform keyboard control of a domain name identifier becomes possible.

1.2 Author’s Disclaimer

The author is not associated in anyway either as a member or as a consultant with any of the above mentioned standards, or standard bodies, or any other commercially operated entities and can not be responsible for any consequence raised from either inclusion or exclusion of any names mentioned
1.3 Terminology

The key words "MUST", "SHALL", "REQUIRED", "SHOULD", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in [RFC2119].

Examples in this document use the notation from the Unicode Standard [Unicode3] as well as the ISO 10646 names. For example, the letter "a" may be represented as either "U+0061" or "LATIN SMALL LETTER A". Examples also use octet notation from national code exchange standards to represent a Unicode character, such as "5167".

1.4 IDN summary

IDN-Map is a basic international code exchange table to support interoperability across various existing clients and local servers on the Internet. It accomodates existing user requirements, engineering feasibility, DNS stability and security, and provides a bridge from existing user platforms to new applications based on the table of Unicode standard.

2. IDN Standards Code Exchange Table

The character set in UCS is a super set of many national code exchange standards as well as many symbols outside those standards. Vast existing applications built on such national code exchange standards are highly crafted to serve large groups of language specific users [UNAME]. While these existing local standards are not compatible with each other, they are compatible with ASCII, any of its symbols may be expressible with alphanumeral of ASCII characters. Through such an alphanumeral, a mapping between a symbol in a local standard to a code point in UCS is easily achievable.

2.1 Structure of IDN Code Exchange Table

Due to IDN name preparation requirement [IDN req], many of the symbols used in common names need to be normalized and canonicalized [nameprep] before they can be used as IDN identifiers. Thus the IDN Code Exchange Table has two columns to satisfy such a primary requirement, and the third column is the corresponding T-ACE identifier for each UCS IDN identifier of the primary language users of those identifiers. The three columns are called Unicode-full-section Unicode-primary-fold and ACE-primary tagged, and short as U-s, U-p, and A-p respectively as in the following example:

<table>
<thead>
<tr>
<th>U-s</th>
<th>U-p</th>
<th>A-p</th>
<th>U-j</th>
<th>A-j</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+0041</td>
<td>U+0061</td>
<td>a</td>
<td>(Latin Letter A case folding)</td>
<td></td>
</tr>
<tr>
<td>U+2fc2</td>
<td>U+2ee5</td>
<td>yv2</td>
<td>(Han character fish for Chinese case folding)</td>
<td></td>
</tr>
</tbody>
</table>

The three columns define a primary IDN code exchange table, and referred as "IDN Primary Map" here after. When there are more than one spoken language users for the same UCS codepoints, one or more secondary languages are added to the primary map. For example, a Japanese Kanji "Fish" corresponding with the same UCS code point U+2fc2 is added to the above map, then:

<table>
<thead>
<tr>
<th>U-s</th>
<th>U-p</th>
<th>A-p</th>
<th>U-j</th>
<th>A-j</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+0041</td>
<td>U+0061</td>
<td>a</td>
<td>(Latin Letter A case folding)</td>
<td></td>
</tr>
<tr>
<td>U+2fc2</td>
<td>U+2ee5</td>
<td>yv2</td>
<td>(Han character fish)</td>
<td></td>
</tr>
</tbody>
</table>

The U-j column equally can be U-k, for Unicode tagged as Korean, a Hangul code point may be there just as well. Or Korean can be additional two
columns added to the secondary map.

2.1.1 IDN-Map that Never Shrinks

It is REQUIRED that a IDN Primary Map contains a column of all permitted symbols, sorted by UCS code points, used in an IDN names, and it is called the "UCS input codepoint". It is also REQUIRED that a IDN Primary Map contains a column of corresponding IDN identifier symbols, called UCS-folded codepoints, and a column of corresponding ASCII symbols permitted by [STD13] to be used for DNS identifiers, called DNS-codepoints. Data items in a IDN Primary Map MUST NOT be removed, MUST NOT be altered in anyway whence it is deployed.

It is REQUIRED that after a secondary language added onto an IDN primary map, the items in such an addition MAY NOT be removed, MAY NOT be altered in anyway whence it is deployed. The additional columns of a secondary language is called IDN secondary map, and each item in a secondary map MUST correspond with its primary map entry in associated UCS input codepoints.

2.1.2 Equivalent Symbol Set Mapping

Equivalent Symbol set of a script is common and it is important to identify such equivalency in the context of IDN identifiers on naming the same entity with semantically equivalent symbols especially when IDN provides far more potential use for symbols from mixed scripts. IDN-map is a convenient vehicle to carry out equivalent symbol set by providing more referencing columns, called Equites Map and shorted as U-e, to the IDN Primary Map, as such:

<table>
<thead>
<tr>
<th>U-e</th>
<th>U-s</th>
<th>U-p</th>
<th>A-p</th>
<th>U-j</th>
<th>A-j</th>
</tr>
</thead>
</table>
| U+0410 | U+0041 | U+0061 | a | (Latin Letter A case folding)
| U+??? | U+2fc2 | U+2ee5 | yv2 | U+2fc2 | uo (Han character fish) |

or in IDN Primary Map format:

<table>
<thead>
<tr>
<th>U-s</th>
<th>U-p</th>
<th>A-p</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a'</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a“</td>
<td>a</td>
<td>a</td>
</tr>
</tbody>
</table>

Access support to Equites Map is NOT RECOMMENDED for applications discussed in this document, since the focus here is for the ease of the largest common denominator code exchange.

2.2 Access of IDN Code Exchange Table

Many access method can be supported with IDN code exchange map, they are the universal access and local access, where a local access MAY be deprecated in the future when universal access becomes direct global access to every one in particular local area.

2.2.1 Universal Access

The IDN Primary Map offers two types of access: 1) Unicode input through a screen selection or URL buffer and receive a DNS codepoint in its primary language users’ favor, and is called "idn-umap"; 2) to access through a DNS codepoint and retrieve its corresponding UCS codepoint for display.

The IDN maps sorted by codepoints in particular column are called IDN access maps, and the access through primary DNS compatible codes is called the IDN Primary Access Map, and is called IDN Tagged Primary Access Map, for subsets of IDN Primary Access Map. For example, UCS CJK section in IDN Primary Access Map is called IDN Chinese Primary Access MAP, or
"idn-zh-pmap" for short. It is REQUIRED that the IDN Tagged Primary Access Maps are NOT overlap with each other in terms of UCS codepoints.

There is also the potential in over fragmenting IDN Primary Map, and causing unnecessary processing overhead for both machine time and user frustration. Reasonable studies are REQUIRED in defining Primary Access Maps to facilitate different language groups using the same Primary Access Maps, such that Primary Access Maps are not fragmented into excessively small maps.

The DNS codepoint access map for a secondary language user is called IDN Tagged Secondary Universal Access Map. Thus a Korean universal access map is named as "idn-kr-amap".

IDN Universal Access Maps MUST be updated when IDN primary map is updated.

### 2.2.2 Local Access

Many existing local display standards are the basic code points in the client system and local server systems. They are limited to highly efficient set of operations for the end users as well as processes for the local servers. To facilitate end users for the speed of IDN access as well as compatibility with existing applications, it is RECOMMENDED that an IDN code exchange table includes applicable local display standards corresponding with each applicable codepoints in UCS. Taking the example from Section 2.1:

\[
\begin{array}{ccccccc}
U+0041 & U+0061 & a & & (Latin Letter A case folding) \\
U+2fc2 & U+2ee5 & yv2 & U+2fc2 & uo & (Han character fish) \\
\end{array}
\]

after including local code standards, it becomes:

\[
\begin{array}{cccccccccc}
0 & 2-1 & 2 & 2+1 & 2+2 & 6-1 & 6 & 6+1 & (Column number) \\
U+0041 & U+0061 & a & & (Case folding) \\
U+2fc2 & U+2ee5 & yv2 & 5167 & b3bd & U+2fc2 & uo & ??? & (Han character fish) \\
\end{array}
\]

Where
- G-p: GB standard in primary language of codepoint U+2fc2
- B-p: Big5 standard in primary language of codepoint U+2fc2
- J-j: JIS standard in Japanese language of codepoint U+2fc2

The Column number in the first row are identified with a language tag discussed in Section 2.3.1. The column# with "r" are local access maps. They are called idn-zh-lmap-gb, idn-zh-lmap-b5 and idn-ja-lmap-ji respectively, and their column number is an off-set index from its tagged ACE column number.

It is RECOMMENDED, that when a local display code standard is not used for any legitimate reasons, it MAY be deprecated from IDN code exchange table, and any new application based on the IDN-map MAY NOT depend on local access maps.

### 2.2.3 Summary of IDN Maps

A list of IDN maps using the Column header in example in section 2.2.2, where (S) indicates the sorted column with the map naming:

**Full maps:**

\[
\begin{array}{cccccccccc}
0 & 2-1 & 2 & 2+1 & 2+2 & 6-1 & 6 & 6+1 & (Column number) \\
0-3 & 0-2 & 0-1 & 0 & 2 & 6 & (Column number) \\
\end{array}
\]
Tagged section maps:

idn-la-pmap U-s U-p A-p(S) (Latin section)
...  
...

Local access maps:

idn-zh-lmap-b5 U-s U-p A-p G-p B-p(S) U-j A-j J-j (Chinese BIG5 access)
...

2.2.4 Syntax of IDN Maps

The syntax of IDN maps MUST conform in full with definition specified in Section 3 of [Version]. In addition, a third field of the values is specified as the language tagged, [STD13] conforming IDN names, or DNS identifiers.

It is further specified, if any fields in a line is empty within a given language tagged code block, a field separator ";" MUST be used to maintain data fields alinement.

It is REQUIRED that each line of IDN-map is treated in its entirety in sorting and its columns MUST consistent with its column number sepcified in its full map, idn-umap.

A separated text file, and is proposed to be named as "idntag-xy.txt", to specify particular Unicode blocks applicable to a particular language tag and its data fields or column number definition. More discussion regarding the IDNTAG file is in the next section.

3. Version control and Language tags of IDN Code Exchange Table

UCS character set is an open set, there are possible updates to let in new scripts as well as new individual characters. There are also possible longer preparation time required for certain subsets to be deployed, as well as possible increased user demand for mixed script use in the future. Language tag defined by [ISO639][RFC 3066] MUST be used as a flag in 1) defining a ready to serve a language group as apposed to unspecified language group such as mathematic "language", 2) ready to serve script range in terms of Unicode blocks, 3) ready to find corresponding mneumonic ACE for a UCS codepoint and vice versa.

3.1 Language Tags

A language tag is define by [ISO639-2/T] and [RFC 3066], and it MUST be prepended to a DNS name label and followed by a hyphen "-" in the form of "xx-". A tag MUST have at least one non-zero Unicode block, R1, as its associated script range, defined by a triple: (start-point, end-point, Column# of T-ACE in IDN-Map), or (0001, ffff, n), where start-point <= R1 >= end-point of Unicode code points; and column# MUST be an positive interger, n, where n-1 is the tagged Unicode folded column, and n+1, n+2, ... , m, are column# of the local display standards of the language tag.

The first code block of a script range is the primary range of a language
It is REQUIRED that none of the primary ranges of language tags are overlap for feasible covering of error checking and consistent assignment of T-ACE value. It is also RECOMMENDED to test for operational complexity before increasing its associating number of blocks, or to expand its script range. It is REQUIRED to register a language tag with IANA and its associated script range whenever it is modified. The repertoire of the registered tags and their script ranges is called IDNTAG file here after.

### 3.2 Language Tag File Format

IDNTAG file has a consistent format specified in [Version] Section 3, that is:

- one language tag per line
- lines separated by CR/LF
- each field in the line separated by ";"
- each subfield in the line separated by ","
- the third subfield of the first triple field in a line is a constant for all primary language tags for ease of maintainess.

such that the IDNTAG file takes on the form:

```
tag-name; version#; block-1; block-2; block-3;...
```

where each block has a three subfields, specifying the starting and ending codepoint of a block in form of Unicode hexadecimal, and an interger as the number of T-ACE column in IDN map. For example:

```
tag1;1.0;HHHH,HHHH,2;HHHH,HHHH,6;HHHH,HHHH,5;
tag2;1.0;HHHH,HHHH,2;HHHH,HHHH,5;
...
```

### 3.3 Identification of a Tag of an Input String

An IDN address in URL format may be in any mixed scripts, but all the characters of an IDN label MUST be in the same script range of one language tag. This conformity ensures correct treatment of an IDN label by any URL parsers, and minimizes confusion codepoints among different scripts. To use mixed scripts in one IDN label is NOT RECOMMENDED for an early deployment of IDN.

#### 3.3.1 IDN Tag File Interface

An IDN label can be an arbitrary byte stream in IDN-Map permitted display code standards ([ISO10646](#) and others to be decided), and a four parameters for such an interface to IDNTAG file is defined as:

```
stat = find-tag(input, tag-file, input-std, tag-rec)
```

where find-tag MUST have four parameters:

- input: a string in input standard byte stream,
- input-std: one of the code exchange standard permitted in IDN-map, including (UCS, USASCII, GB, BIG5, JIS, KSC, ISC ...)
- idntag-file: tag definition file specified in Section 2.3.2.
- tag-rec: a buffer for returning triples as defined in Section 2.3.1.

stat: status of the search including (ERROR, USASCII, UCS, ALPH, CONS, CJK, NO-TAG, LOC), and discussed in Section 3.3.3.

Find-tag portocol is REQUIRED before each access to IDN-Map.

#### 3.3.2 IDN Language Tag Identification Protocol

The above find-tag protocol is REQUIRED to include the following actions
and the actions are performed in the following orders:

1) to identify tag prefix of an DNS label and returns a tag's triples;
2) to identify an ASCII DNS label, if it is conforming to [STD13], and assign USASCII to the tag value, and return USASCII to tag status;
3) to assign a tag, if the input standard has a known language tag, for example, input standard JIS implies language tag "ja", returns tag triples;
4) be defaulted to UCS and checking for script range error. It is RECOMMENDED that at least two of the input Unicode codepoints to be checked for more accurate tag identification. Inconsistent tag values between the two check points, the more specific value MUST be returned, and a corresponding tag triples MUST be returned;
5) to assign a language tag status to the protocol, when no applicable tag is found, and a prohibited codepoint is not encountered, a NO-TAG value MUST be returned.

3.3.3 IDN Language Tag Identification Status Protocol

Tag identification is RECOMMENDED to use at least two of input codepoints, for higher accuracy and a two-step classification as well: one for its script group, the other for the script within the group. The first step is to identify script group. Since scripts may be treated in three different groups: alphabet, consonant and syllabic or character-based systems. The three groups is reflected by the following code blocks in UCS as shown below:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>From:</td>
<td>0020</td>
<td>2e80</td>
</tr>
<tr>
<td>to:</td>
<td>052f</td>
<td>d7af</td>
</tr>
<tr>
<td>include:</td>
<td>Latin</td>
<td>CJK</td>
</tr>
<tr>
<td></td>
<td>Greek</td>
<td>Armenian</td>
</tr>
<tr>
<td></td>
<td>Cyrillic</td>
<td>CJK</td>
</tr>
<tr>
<td></td>
<td>IPA</td>
<td>CJK</td>
</tr>
<tr>
<td></td>
<td>Vietnamese</td>
<td>CJK</td>
</tr>
<tr>
<td></td>
<td>Thai</td>
<td>CJK</td>
</tr>
<tr>
<td></td>
<td>Lao</td>
<td>CJK</td>
</tr>
<tr>
<td></td>
<td>Tibetan</td>
<td>CJK</td>
</tr>
<tr>
<td></td>
<td>Yi</td>
<td>CJK</td>
</tr>
</tbody>
</table>

Some cultures often use more than two scripts within the same group, such as Japanese, but rarely using another script especially from a different group. The three different groups also reflect different processing consideration as well.

Scripts in Alphabet group are frequently used by different language users who may mix two or more different spoken language names using the same script. Also, alphabet has semantically equivalent two sets of symbols: uppercase and lowercase letters, which can be folded under [nameprep] canonicalization. The main treatment issue is to consider mixed symbol use, for different language groups, for example, an Azerbaijain may wish to switch between Latin and Cyrillic at easy.

The majority scripts in Consonant group are one language per script, where many of the symbols from different scripts are look-alike but have unrelated values. However, when such a look-alike symbol in its own script contexts, its value is unambiguous. IF the script is correctly identified, potential symbol confusion would be resolved. In this group, more language tag identification care should be given than members of other script groups.
Treatment of Character based scripts is largely a uniqueness of characters’ indices. The issue is more contentious if a character T-ACE collides with another T-ACE on a different character. Also, due to its meagre size of symbols, its T-ACE index system has to be easily mastered and to be sorted for fast access [StepCode]. The main issue in IDN-Map is to identify character equivalent sets, and reduce the number of applicable IDN identifiers by 1) limiting the applicable IDN input code points to Plane 0 of Unicode table, 2) assigning one IDN identifier from each semantically equivalent character class suggested by [CJK], [tsconv].

3.3.4 Summary of IDN Language Tag Status Protocol

The three major script groups are status as ALPH, CONS, and CJK, as they are mentioned in Section 3.3.1, and 3.3.3. It is suggested that language tags are fall into the same script groups, MAY be treated with the same language specific normalization and presentation methods discussed later in Section 4.3 of this document, to reduce implementation complexity.

IDN Language Tag status also has

NO-TAG: Unicode input code points without primary language tag defined,
ERROR: prohibited UCS input code points [nameprep],
LOC: code points of local standards permitted in IDN-map other than Unicode;
USASCII: [STD13] compliant input string.

4. Interface with IDN Code Exchange Map

A uniform interface with IDN map is specified for interoperability among different clients and local server systems, and feasible upgrade of language specific modules associated to an individual language tag. These language tag specific modules are called "language tagged procedures".

4.1 Language Specific Modules

A spoken languages is expressed with specific symbols grouped into a corresponding script, which may be scattered in different UCS blocks. Each script has its own methods in manipulating its symbols, in decomposing a symbol into parts, in selecting a symbol from an equivalent symbol set, in combining symbols into a string, as well as in presentation of a string on a screen. However, each language has each own systematic way to treat its script, some processes can be captured in simple procedures, others have to be treated on an individual basis, and many variations are in between. It is RECOMMENDED that reasonable studies are given to each language to classify script treatment model, and a cost vs. benefit analysis in select a long term script specific processing protocol to be embedded in IDN language specific modules. It is RECOMMENDED that processing speed and simplicity of its implementation takes the highest priority in such a decision.

Two levels of script specific processing are supported with IDN-Map structure. The lowest level is the language tagged IDN map in favor of the primary users of a script (Section 2.1), where a simple code equivalence from input to an IDN identifier can be assigned, and is referred as canonicalization in [UTR21], [tsconv], [jpchar], [hangeul]. The second level is IDN label normalization and presentation.

4.2 Script Specific Canonicalization

The first level of script specific canonicalization have been addressed in [nameprep], [tsconv], [jpchar], [hangeul], [bidi], [UTR21], [CJK], where a mechanism of folding by Domain Name registration services and at client site for the purposes of preventing confusing allocations CJK Domain Names or the likes take much higher priority in domain name services.
For local server based deployment of IDN, a partial solution of recover the registered codepoints MAY be achieved by specifying the presentation of IDN use prefolded form for all of the names. For example, "JOES-Pizza" is folded to "joes-pizza", and recoved to "JOES-PIZZA" when the user has such a desire.

Another complete recovering solution would involve a different server transport of the original registered form, where a supporting mechanism is discussed in [UNAME} and is used in CJK specific procedures in Section 4.3.1 and 4.3.2.

Uniform interface to IDN map has one procedure with 5 parameters:

\[
\text{idn-folding(input-list, input-std, tag-rec, output-std, output-list)};
\]

where

input-list is the normalized and error checked codepoints \([bidi][UAX15]\),
input-std is the code standard of the normalized input label (Sec.3.3.1),
tag-rec is the returned tag triples from find-tag protocol (Sec.3.2),
output-std is the requested code standard, same as input-std,
output-list is a list of all the codepoints retrieved from IDN Map in output-std;
and

input-std and output-std are couples of intergers in the form of \((a,b)\),
where the integer, \(a\), is the input-std (Sec.3.3.1) and the second integer, \(b\), is the off set number of columns from corresponding T-ACE column number (Sec.2.2.3).

4.3 Language Specific Normalization and Presentation

The second level of script specific processing have been addressed in [IDNA], [icdn], [UAX15], [UAX9] and [bidi] are referred as normalization procedures, and presentation procedures.

Normalization is to break an input string into a list of UCS codepoints in input code standard. Presentation is to combine a list of UCS codepoints into a string in output code standard. Presentation may joint certain symbols between UCS codepoints or rendering the order of UCS codepoints’ presentation as a string. Normalization MUST reverse all the renderings made by its corresponding presentation procedure on a label string when it break a string into a list of UCS code points. When input is an ACE string similar processes are calles "fitting" and decompose". The relations are:

\[
\begin{array}{ccc}
\text{input} & \text{Processes} & \text{output} \\
\text{UCS} & \text{normalize-->fitting} & \text{ACE} \\
\text{ACE} & \text{decompose-->present} & \text{UCS} \\
\end{array}
\]

For convenience, these procedures are proposed to be named with the exact language tag defined in IDNTAG file in the name, such that a language tagged normalization procedure is named as "idn-XY-normalize", where "XY" represents the language tag of associating procedure. Following the same convention that "idn-XY-present", "idn-XY-fitting", "idn-XY-decompose" would be the names for respective DNS name decompose procedure and IDN name presentation procedures. For example, "idn-zh-present" is the language tagged IDN label presentation procedure for Chinese.

Two language specific script treatment procedures are REQUIRED for each language tag registered: 1) Normalize and 2) Present, and two additional T-ACE specific script treatment procedures 3) Fitting, 4) Decompose are RECOMMENDED for non-alphabet languages. It is also RECOMMENDED that a
NO-TAG general compressive ACE [AMC] is registered as compress and decompress procedures corresponding with Fitting and Decompose procedures with IANA. It is REQUIRED that when a language tag is registered with IANA, the associated script specific procedures to be registered at the same time.

4.3.1 Language Tagged Normalization and Input Error Checking

The find-tag interface gives the legal search range for error checking and normalization process to insure all the codepoints in input IDN label are legal IDN codepoints, which SHOULD NOT be rejected by IDN Map. The returned list of UCS codepoints MUST be checked for such an error, to prevent illegal IDN codepoint slip through and burden its following search in IDN-Map. The normalization protocol is:

\[
\text{stat}=\text{idn-XY-normalize}(\text{input}, \text{input-std}, \text{tag-rec}, \text{input-list}, \text{err-report})
\]

It is REQUIRED that each language tagged normalization procedure perform:
1) check for disallowed input-std,
2) check for disallowed codepoints in its script range,
3) normalize input string to IDN-Map allowed input codepoints,
4) return input-list with one UCS codepoint per record,
5) report any errors.

A similar protocol for

\[
\text{stat}=\text{idn-XY-decompose}(\text{input}, \text{USASCII}, \text{tag-rec}, \text{input-list}, \text{err-report})
\]

It is RECOMMENDED that each T-ACE decomposition procedure perform:
1) check zonefile for cached IDN label
2) check for Non-ASCII input string for transport corruption,
3) check label length, if it is up to the maximum, request for the original registered IDN label from registrar,
4) strip language tag,
5) decompose input string to IDN-Map permitted UCS code points,
6) return input-list with ACE for each UCS codepoint per record,
7) report any errors.

4.3.2 Language Tagged Presentation and Preserving Character Boundary

When idn-fold protocol returns a list of output UCS codepoints, a presentation process checks correctness of output codepoints and combines these codepoints into a display string. If output codepoints contain errors, presentation procedure SHOULD report an error, and request the original IDN display codepoints to be send, and make its best effort in display the current IDN string. The presentation protocol is:

\[
\text{stat}=\text{idn-XY-present}(\text{output-list}, \text{output-string}, \text{err-report})
\]

It is RECOMMENDED that each language tagged presentation procedure perform:
1) if a codepoint contain an error, request for the original registered IDN label from original registrar,
2) reverse randerings made to a string by normalization procedure,
3) arrange string display order/direction,
4) concatenate output-list to output label and return the output label,
5) report any errors.

A similar protocol for

\[
\text{stat}=\text{idn-XY-fitting}(\text{output-list}, \text{output-name}, \text{err-report})
\]

is to put in necessary separtors for easy decomposing, and make it certain the encoding length fit into limited label space of 63 octets. If the
encoding is over maximum label length, it SHOULD record both input string
and T-ACE name to local zonefile, and compose a DNS identifier from
output-list codepoints.

It is RECOMMENDED that each T-ACE fitting procedure perform:
1) check for total code length, truncate certain tailing ACE to fit into the
label limit length if required,
2) when necessary, put codepoint separator for proper decomposing,
3) concatenate ACE from each UCS code point to an output-name,
4) prepend the language tag to output-name,
5) report any errors.

4.3.3 Special Attention to Mix Scripts

A string mixed with CJK and Kana is Japanese, CJK and Hangul mix is
Korean. However, an all CJK character string MUST presumed to be in the
primary language tag, that is Chinese, and registered as the only IDN name,
unless the registrant requests a second and a third language to access the
same IDN name. In this case, there could be more than one DNS label to be
maintained by the registrant, and the IDN-Map becomes an automatic name
translation agency.

Tag identification of an arbitrary input string proposed in find-tag
protocol is an language indicator at its best. More careful check should
be given in normalizing and error checking procedure. For example, the
Chinese tagged normalizing procedure, idn-zh-normalize, MUST check all input
points to be certain about the correctness of returned value from find-tag
procedure, and alter when it is necessary. It SHOULD identify a CJK-Kana
mix as Japanese tag, and CJK-Hangl mix as Korean tag.

4.4 Language Tagged IDN Label Conversions

The primary IDN label conversions are from UCS to [STD13] and vice versa.
A backward compatibility utilitary support is also given to a limited set of
local standards. Uniform IDN interface to applications is concured by IETF
treat any possible input string with the same procedures, and divert
language specific requirement to language tagged procedures at fixed points
of IDN label conversions.

The uniform IDN interface to applications is proposed to be:

    idn-label(input, input-std, tag-file, zone-buff, idn-name, output-std);

where
input: IDN label in input-std,
input-std: any IDN permitted code standard (Sec.3.3.1),
tag-file: IANA distributed IDNTAG file (Sec.3.2),
zonelfile: optional local registered domain name file for servers [UNAME],
or cache at a client site,
idn-name: output of converted input in requested output-std,
output-std: requested output form in any IDN permitted code standard.

In addition a localized zonefile search procedure SHOULD be supplied if a
zonefile is applicable.

4.4.1 Code Conversions Supported by IDN-Map

Idn-label protocol recognizes two code standards: UCS and ASCII by default.
Any other permitted code standards MUST be specified as parameters. The
code conversion direction is specified in the following matrix.
Input-std to output-std implementation matrix:

<table>
<thead>
<tr>
<th>in\out</th>
<th>U-i</th>
<th>U-p</th>
<th>ACE</th>
<th>ASCII</th>
<th>G</th>
<th>B</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-i</td>
<td>-</td>
<td>fold</td>
<td>DNS</td>
<td>-</td>
<td>disp</td>
<td>disp</td>
<td>disp</td>
</tr>
<tr>
<td>U-p</td>
<td>record</td>
<td>-</td>
<td>DNS</td>
<td>-</td>
<td>disp</td>
<td>disp</td>
<td>disp</td>
</tr>
<tr>
<td>ACE</td>
<td>record</td>
<td>regist</td>
<td>pass</td>
<td>pass</td>
<td>disp</td>
<td>disp</td>
<td>disp</td>
</tr>
<tr>
<td>G</td>
<td>record</td>
<td>fold</td>
<td>DNS</td>
<td>-</td>
<td>-</td>
<td>disp</td>
<td>disp</td>
</tr>
<tr>
<td>B</td>
<td>record</td>
<td>fold</td>
<td>DNS</td>
<td>-</td>
<td>disp</td>
<td>-</td>
<td>disp</td>
</tr>
<tr>
<td>J</td>
<td>record</td>
<td>fold</td>
<td>DNS</td>
<td>-</td>
<td>disp</td>
<td>disp</td>
<td>-</td>
</tr>
<tr>
<td>ASCII</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>pass</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

where
- U-i UCS input
- U-p UCS folded in primary language
- ACE T-ACE form
- G,B,J permitted local code standards
- record used for registration font or trademark records
- regist for registration conflict matching
- fold canonicalization case folding
- DNS obtain DNS identifier
- pass pass by, no process
- disp local client backward compatible display
- - prohibited

From observation of the matrix, it is clear, that the conversion is based on input code standard. If the input and output are all ASCII, then output is ASCII without any further delay, which is compatible with current DNS operation.

4.4.2 Input and Output Format Request

Considering that idn-label protocol may be installed on a client site, the input and output request specification may contain errors due to variety of inconsistent site configuration, smooth handling of such errors is an important part of idn-label protocol.

Input-std to output-std default case matrix:

<table>
<thead>
<tr>
<th>in\out</th>
<th>U-t</th>
<th>U-n</th>
<th>ACE</th>
<th>ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-t</td>
<td>-</td>
<td>-</td>
<td>ACE</td>
<td>-</td>
</tr>
<tr>
<td>U-n*</td>
<td>-</td>
<td>-</td>
<td>ACE</td>
<td>-</td>
</tr>
<tr>
<td>ACE</td>
<td>UCS</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ASCII</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>pass</td>
</tr>
</tbody>
</table>

where
- U-t UCS code with tag identified
- U-n UCS code with NO-TAG identified, *also any input-std error case
- ACE identified ACE format
- ASCII [STD13] with no tag, or with "us-" tag added by zone masters
- ignored case
- pass by without any processing

It is proposed that the tag "us-" is reserved for a name part which consists exclusively of characters that conform to the hostname requirements in [STD13], as an optional language tag. If an all ASCII label in [STD13] or a "us-" prepended to a name, and the output standard is not specified, or is specified as USASCII, then the input name MUST NOT be converted at all. This absolute requirement prevents:
1) double encoding from a client of user keyboard input and a server provider;
2) messing up existing registered domain names;
3) interfering with registered glyphs with more than one phonetic standard, such as Hanja and Kanji in CJK script.
If the input string consists only of characters that conform to the hostname requirements in [STD13], and with a prefixed language tag, and the output standard is NOT USASCII, the RECOMMENDED output defaults to UCS folded, column #1, which is the universal base support. This recommendation is to provide a friendly presentation for end user configuration ignorance.

When there is no tag on a non-ASCII input string, then it is going through script identification, prohibited characters filtering, canonicalization, case-folding, as defined in [nameprep] and is treated with find-tag process.

If its output-std is not specified or specified with inconsistence, then the USASCII is assigned as the default output-std for any non-ASCII input.

All the rest input and output code standards MUST be explicitly specified for any conversion requests to be honoured.

4.5. Uniform Idn-label Protocol

The Idn-label protocol is summarized in a C language format, with some of the parameters and details omitted.

```c
idn-label(input, input-std, tag-file, zonefile, idn-name, output-std)
{
    flag = find-tag(input, tag-file, input-std, tag-rec);
    tag = get-tag(tag-rec);

    /* Part 1: Name preparation, normalization and error checking */
    switch (flag)
    {
        case ERROR:  return(ERROR);
        case USASCII:                   /* input ASCII */
        {
            switch (tag)
            {
                case NIL: return (idn-name = input);   /* ASCII passby */
                case US: return (idn-name = input);    /* ASCII passby */
                case AMC:                              /* General ACE[AMC] */
                    {idn-amc-decompress;
                      return(idn-name)}              /* Finish */
                case ZH: idn-zh-decompose;             /* T-ACE decompose */
                case JA: idn-ja-decompose;
                case KR: idn-kr-decompose;
                ...
                default: return ("unimplemented tag ERROR");
            }
            if (output-std not permitted)          /* Output request check */
                output-std = UCS;
        }
        case NO-TAG:                              /* General UCS input */
        {
            switch (flag)
            {
                case ALPH: idn-alph-normalize
                case CONS: idn-cons-normalize
                case CJK: idn-cjk-normalize
            }
            if (output-std ERROR)                /* Output request check */
                output-std = USASCII;
        }
    }
}
```
default: /* script range found */
{
switch (tag)
{
    case zh: idn-zh-normalize
    case ja: idn-ja-normalize
    case kr: idn-kr-normalize
    ...
}
if (output-std not permitted) /* Output request check */
    output-std = USASCII;
}

/* Above normalizing protocol:
stat&idn-XY-normalize(input, input-std, tag-rec, input-list, err-report)*/

if error (stat) /* Input error checked */
{
    fprintf(stderr, "%s %s", input, err-report);
    return (ERROR);
}

/* Part 2: Canonicalize and Code exchange */
idn-folding(input-list, input-std, tag, output-std, output-list);

/* Part 3: Present and Fitting */
switch (output-std)
{
    case USASCII: /* output ACE */
        {
            if (flag = NO-TAG) { stat&idn-AMC-compress; tag=AMC;}
            switch (tag)
            {
                case zh:
                    stat&idn-zh-fitting(output-list, idn-name, err-report);
                case kr:
                    stat&idn-kr-fitting(output-list, idn-name, err-report);
                case XY:
                    stat&idn-XY-fitting(output-list, idn-name, err-report);
            }
            concatenate{ tag, idn-name}; /* prepend tag to ACE*/
        }
    case UCS: /* output UCS */
        {
            switch (tag)
            {
                case AMC:
                    switch (flag)
                    {
                        case ALPH: idn-alph-present;
                        case CONS: idn-cons-present;
                        case CJK: idn-cjk-present;
                    }
                case kr: idn-kr-present;
                case ja: idn-ja-present;
                case XY: stat&idn-XY-present(output-list, idn-name, err-report);
            }
        }
    case other-output:{}/* output other standard */
}
}
5. Preferred Embodiment of IDN Code Exchange Map

Three applications are suggested for client, server and general public.

5.1. Client Application

Uniform Idn-label Protocol of Section 4.4 is one of the preferred embodiments of IDN-map discussed to provide consistent IDN client interface across any language installation. Using Idn-label interface, a basic URI cut and paste operation may be implemented:

URL cut and paste, then send:
Loop for all labels

- Get IDN label from URL buffer,
- Call Idn-label, receive ACE label,
- replace IDN label with ACE label
  until end of URL
}
send URL.

5.2 Server Application

The most important embodiment of IDN-Map is in IDN Domain Name registration process to check for name conflict and trademark search, where trademarks in Han characters is common practise. The following prototype demonstrates such an embodiment.

IDN registration as an example for server application:
1) get wish-name,
   call Idn-label(wish-name), receive T-ACE-label.
   examining T-ACE-label, if bad go to 1).
   send T-ACE-label for DNS match, bad go to 1).
   good go to 2)
2) call Idn-label(T-ACE-label), receive IDN-name.
   examining IDN-name, if bad, go to 1).
   send IDN-name for IDN match, if bad go to 1).
   good, go to 3).
3) Register IDN-name, T-ACE-label in zonefile [UNAME].

5.3. Implications of Deployment of IDN-Map

IDN-Map is a feasible tool for many, for example, a third application has been suggested to use the IDN-map as a general input encoding exchange module to be called from any applications. If it is implemented then a librarian may use a keyboard with existing input software to access a particular CJK character, C, in UCS Plane 0, and retrieve a C' from Plane 1, or C" from Plane 2.

A flexible tool always brings its cons with it. From technical area, more scrutiny has to be placed for each equivalent symbol to be mapped into its equivalent code point, and each T-ACE has to be checked for mnemonic pros, simple logical assignment to ensure consistence and uniqueness. Also, it introduces more policy decisions, for example, an all CJK character trademark registrant may have to registrate in three languages to ensure the legitimacy of the trademark. After all, a useful tool is to let its user to make decisions.

6. Security Considerations

Much of the security of the Internet relies on the DNS. Thus, any change to the characteristics of the DNS can change the security of
much of the Internet. IDN-Map makes no changes to the DNS itself.

7. Internationalization considerations

The International code exchange table will provide convenience for many international application development.

8. Acknowledgements

The special comments which have contributed to improve this document were received from Li Ming Tseng as well as many other people from the working group.

9. IANA Considerations

This document requires IANA action for availability of language tag, and registration for each tag and associated language specific processing procedures.

10. References


[GB] China national code exchange standard.


[IDN] "IETF Internationalized Domain Names Working Group", idn@ops.ietf.org, James Seng, Marc Blanchet


[IDNReq] Zita Wenzel and James Seng, "Requirements of Internationalized Domain Names", draft-ietf-idn-requirements. May 2001.)

[ISCI] Indian Standard Code for Information Exchange


[RFC2231] Email tag


11. Authors’ Contact Information

Liana Ye
YsD ISG
2607 Read Ave.
Belmont, CA 94002, USA.
(650) 592-7092
liana.ydisg@juno.com