Internet Draft   Paul Hoffman
draft-hoffman-idn-reg-02.txt   IMC & VPNC
October 16, 2003
Expires in six months
Intended status: Experimental

A Method for Registering Internationalized Domain Names

Abstract

This document describes some suggested practices for registering internationalized domain names (IDNs) in a zone. Before accepting registrations of domain names into a zone, the zone's registry should decide which codepoints in the Unicode character set the zone will accept. The registry should also decide whether particular characters in a registered domain name should cause registration of multiple equivalent domain names; these domain names might be added to the zone or blocked from registration. This document also describes how to handle character variants in registering IDNs, and how to publish tables that list the character variants.

1. Introduction

IDNA [IDNA] specifies an encoding of characters from the Unicode character set [UNICODE] which is backwards-compatible with the current definition of hostnames. This implies that domain names encoded according to IDNA will be able to be transported between peers using any existing protocol, including DNS.

IDNA, through its requirement of Nameprep [NAMEPREP], uses tables that are based only on the characters themselves; no attention is paid to the intended language (if any) for the domain name. However, for many domain names, the intended language of one or more parts of the domain name actually does matter to the registry for the names and to users.

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

For some human languages, there are characters and/or strings that have equivalent or near-equivalent usages. If someone is allowed to register a name with such a character or string, the registry might want to automatically associate all the names that have the same meaning with the registered name. The registry can also decide if the names that came from one registration should go into the zone, be blocked from other people registering them, or a combination of these two actions.

This document describes three things:

- suggested practices for describing character variants

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

- a format for publishing a zone's table of character variants

[IDN-CJK] offers a somewhat different proposal to the problem of registration policy. That document uses a different registration philosophy than what is described here, and is focused on a small number of Asian languages.

1.1 Main concepts

[[Need to outline what is presented in the proposal. Include:

- registration bundles keyed on the base registration

- bundles can overlap

- some names are in the zone, others only block

- does not prohibit human processing, but does not encourage it

- tables based on languages
]]

1.2 Shortcomings

This document does not deal with how to handle whois data for associated registrations, and does not deal with registrar-registry protocols. These topics are likely to be of great importance to registries and registrants, and should be dealt with in other documents.

This document deals directly only with variants of single characters, not variants of strings (although variants themselves can be strings). Thus, the methods described here is not be sufficient to help all languages. Registries which cover languages where it would make linguistic sense to create variants from strings should define their own rules for doing so.

The procedures described here do not take into account mapping that is dependant on the position of characters in a domain name. Many languages (such as Hebrew and Greek) have rules that would cause different variants to be used based on whether a character appears at the beginning or end of a word, or whether a character appears next to a specific character. Adding rules for these kinds of mappings are possible, but difficult. Not only would the table format need to be expanded to deal with positional variants, the order in which the characters are tested for whether they create variants would also have to be specified.

1.3 Terminology

Characters in this document are given as their Unicode codepoints on U+xxxx format or with their official names.

The following terms are used in this document.

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

This document discusses characters that have equivalent or near-equivalent characters or strings. The "base character" is the character that has one or more equivalents. The "variant(s)" are the character(s) and/or string(s) that are equivalent to the base character. Note that these might not be true equivalent characters: a base character might have a mapping to a particular variant character, but that variant character does not have to have a mapping to the base character.

The "base registration" is the single name that the registrant requested from the registry.

A "registration bundle" is the set of all labels that comes from expanding the base characters for a single name into their variants.

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

2. Starting to add IDNs to a zone

There are four primary considerations when adding IDNs to a zone:

- Which characters should be allowed to be registered

- If any of the characters that are allowed to be registered have variants that should affect the registration process

- How registration bundles are created and maintained

- If there are registration bundles, how they will affect the zone itself and future registrations

2.1 Choosing characters that may be registered

A zone has to decide which characters are allowed to be registered. Before IDNA was standardized, the only characters allowed were the ASCII letters, digits, and the hyphen character. With IDNA, that list is much larger.

The first decision for a zone is whether or not they want to allow IDNA-based labels. If not, they can simply prohibit any label that begins with the IDNA ACE prefix "xn--". Zones with this policy can safely ignore the rest of this document.

If a zone decides to allow IDNA-based labels, it needs to decide which characters are allowed to be registered. It further needs to decide which characters are allowed to be in the zone, and which characters can be registered but not appear in the zone.

Some options for what zones will want to include are:

- the ASCII characters plus just enough characters to represent just one language

- just enough characters to represent a small number of languages

- enough characters to represent many languages

- any character allowed by IDNA

The decision on what to include may be influenced by administrative issues for the zone, such as languages that are normally associated with the zone, or agreements that the zone has made with governmental bodies or other organizations. For example, ICANN has a set of rules on how some top-level domains must act with respect to IDNs [ICANN-IDN]. A zone does not need to declare which languages it does or does not allow in the names in its zone, but making such a declaration makes it clearer to registrants what characters the zone does and does not allow.

It is strongly recommended that a registry act conservatively when starting accepting IDNA-based domain names, even if the registry does not use the ideas described in this document. Registries should start with the smallest number of characters as possible to represent the needs of the zone's registrants. If a registry follows the advice in this document, more characters can be added to the zone later, but once characters are labels that are in a zone, they cannot be removed without causing a lot of administrative problems. The most notable problem with making some characters not allowed in names is that a registry could be forced to remove actively-used names from its zone, thereby causing instability for users of the zone and angering the names' owners.

2.2 Choosing variants

The area of character variants is rife with problems. There is no universal agreement about which base characters have variants, or if they do, what those variants are. For example, in some regions of the world and in some languages, LATIN SMALL LETTER O WITH DIAERESIS and LATIN SMALL LETTER O WITH STROKE are variants of each other, while in other regions, most people would think that LATIN SMALL LETTER O WITH DIAERESIS has no variants. In some cases, the list of variants is difficult to enumerate. For example, it has taken years for the Chinese language community to create variant tables for use in IDNA, and the tables are not widely-accepted at the time of this writing.

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

The list of character variants used in a zone should be stable. Although it is possible to add variants for characters later, doing so can cause confusing with registrants.

Of course, zone managers should inform all current registrants when the registration policy for the zone changes. This includes when IDN characters are allowed in the zone the first time, when characters are added later, and when character variant tables change.

In many languages there are two variants for a character, but one variant is strongly preferred. A registry might only allow the base registration in the preferred form, or it might allow any form for the base registration. If the variant tables are created carefully, the resulting bundles will be the same, but some registries will give special status to the base registration such as its appearance in whois databases.

2.3 Creation and maintenance of registration bundles

Another ramification of having variants is that they will cause zones to have bundles of names. Describing registration bundles to typical registrants will be a very difficult task. (Many current registries have a hard time explaining to registrants what they can or cannot do with their single registrations.) It is likely that registrants can better understand this by having the bundle be identified by the base registration.

A registration bundle must be maintained as a single unit. This is not to say that each names in a bundle is treated the same, but that the administration of each name should be done in the context of the entire bundle. Different names in a bundle should not have different administrators.

Adding additional IDN characters to a zone where some or all labels in a registration bundle are resolved in the zone. A registrant who had a single name could become the owner of group of names, and would be expected to manage that group of names according to the zone's policies. Because managing a group of names is inherently more difficult than managing a single name, zone administrators need to avoid creating new rules that would force current registrants to change the way the manage their zones.

2.3.1 Overlapping registration bundles

Depending on how the registry creates its tables, it is possible for registration bundles to overlap, meaning that two different people own rights to a name. This can cause significant problems for the registry in explaining to users what their rights are for names that contain variants of the name they registered.

Clearly, a registrant cannot register a name that already exists as the base registration for another bundle. However, a registrant can register a name which has a variant that exists in the bundle of an existing registration. That is, bundles can have names that are the same, but the zone can never have two different entries for the same name.

The basic registration rule for most zones is "first come, first served unless the registration is misleading". That rule should probably be extended to the names in a registration bundle. That is, the first registrant whose bundle contains a name gets to have rights over that name. Because of this, the registry must associate a registration date with each bundle.

When a second bundle is created that contains a variant name that already exists in an earlier bundle, the registry can inform the new registrant that not all of the names in its bundle are usable in the same fashion. Further, if the registrant of the first bundle allows its registration to expire while the second bundle still exists, the owner of the second bundle gains control over the overlapping names which before were controlled by the owner of the first bundle.

2.4 Choosing how to use variants

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

1) Resolve all labels in the zone, making the zone information identical to that of the registered label.

2) Block all labels other than the registered label so they cannot be registered in the future.

3) Resolve some labels and block some other labels.

In all cases, at least the registered label should appear in the zone. It would be almost impossible to describe to name owners why the name that they asked for is not in the zone, but some other name that they now control is.

2.4.1 Advantages and disadvantages of the options

Option 1 will cause end users to be able to find names with variants more easily, but will result in larger zone files. For some language tables, the zone file could become so large that it could negatively affect the ability of the registry to perform name resolution. If the base registration contains several characters that have equivalents, the owner could end up having to take care of large number of zones. For instance, if DIGIT ONE is a variant of LATIN SMALL LETTER L, the owner of the domain name all-lollypops.example.com will have to manage 32 zones.

Option 2 does not increase the size of the zone file, but it may cause end users to not be able to find names with variants that they would expect. If the base registration contains characters that have equivalents, Internet users who don't know what the base characters used in the registration will not know what character to type in to get a DNS response. For instance, if DIGIT ONE is a variant of LATIN SMALL LETTER L, and LATIN SMALL LETTER L is a variant of DIGIT ONE, the user who sees "pale.example.com" will no know whether to type a "1" or a "l" after the "pa" in the first label.

Option 3 is likely to cause the most confusion with users because including some variants will cause a name to be found, but using other variants will cause the name to be not found. For example, even if people understood that DIGIT ONE and LATIN SMALL LETTER L were variants, a typical DNS user wouldn't know which character to type because they wouldn't know whether this pair were allocating variants or blocking variants. However, this option can be used to balance the desires of the name owner (that every possible attempt to enter their name will work) with the desires of the zone administrator (to make the zone more manageable and possibly to be compensated for greater amounts of work needed for a single registration).

2.4.2 Operational characteristics

With any of these three options, the registry must keep a database that links each label in the registration bundle to the base registration. This link needs to be maintained so that changes in the non-DNS registration information (such as the label's owner name and address) is reflected in every member of the registration bundle as well.

If the registry chose option 1, when the zone information for the base registration changes, the zone information for all the members of the registration bundle must change in exactly the same way. The zone information for every member of the registration bundle must remain identical as long as any of the members of the registration bundle remain in the zone. A registry can keep the zone information for the registration bundle identical using a database, or using DNAME records, or using a combination of the two.

If the registry chose option 2, when the zone information for the base registration changes, the blocked information for all the members of the registration bundle must be identical to that of the base registration, and must remain identical as long as the base registration remains in the zone. A registry can keep the zone and blocked name information for the registration bundle identical using a database.

If the registry chose option 3, it must use an unspecified method to keep the elements in the registration bundle cohesive. Because of the administrative difficulty involved, this option must only be used under carefully-controlled circumstances. Further, the rules for which names in the bundle appear in the zone and which are blocked must be explained to name owners. It is particularly important to explain the ramifications of overlapping registration bundles, if the registry's variant policies allow their creation.

3. Language-based tables

The registration strategy described in this document uses a table that lists all characters allowed for input and any variants of those characters. Note that the table lists all characters allowed, not only the ones that have variants.

Each table is specific to a single language or a specified group of languages. Although a multi-language table can be produced, it may be simpler to keep each table language-specific and only use the table for the language of the desired registration. For example, it is probably easy to create a single table that would handle both Japanese and French; it would probably be much harder to create a single table that would handle both Arabic and Persian.

It is widely expected that there will be different tables for a single language or group of languages created by different people. Many languages are spoken in many different countries, and each country might have a different view of which characters should or should not be considered needed for that language. For example, some people would say that the Latin characters are needed for various Indic languages, while others would say that they are not.

A table that covers a wide variety of languages will probably allow a much wider range of characters to be used in names. At the same time, that table cannot easily use character variants because variants for one language will be different from the variants used in a different language. To handle conflicting variants among languages, the registry can choose to have no variants for any base characters, or can choose to have variants for a subset of the languages that are expressible in the characters allowed.

The set of processing rules for a language has to be carefully crafted so that all expected variants will be created, and no unexpected variants are created. The procedure in this document assumes that the zone uses just one table when registering a particular name, but a set of tables that are searched in a specified order can be treated like a larger table with a processing order.

3.1 Intended language for a registration

In order to use a language-based table for processing, the registry has to know the language of the name being registered. This information could come by asking the registrant, or by the fact that a registry has rules that only allows a single language. However, the requirement of knowing the intended language leads to a very difficult problem: many valid domain names have no inherent language. Examples of domain names that do not have a language include:

- trade names and family names

- names that are acronyms, all-numeric or a combination of the two ("jln", "3000", "jln3000")

- names that purposely have more than one language in them ("neuvo-ramen")

- proper names that are made up ("glowow")

4. Registration procedure

This procedure has three inputs:

- the proposed base registration

- the language for the proposed base registration

- the processing table associated with that language

The output of the process is either failure (the base registration cannot be registered at all), or a registration bundle that contains one or more labels (always including the base registration). As described earlier, the registration bundle should be stored with its date of creation so that issues with overlapping elements between bundles can later be resolved on a first-come, first-served basis.

There are two steps to processing the registration:

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

2) Process the base registration with the CreateBundle process described below.

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

4.1 Human intervention in registration

Some registries will want registration to be completely automatic, that is, with no human intervention. Other registries will want to have human intervention (or at least checking) of registrations. For example, if a registry has a rule that registration cannot have harmful words, that registry needs to have a human check each registration. Another example where human intervention would be needed is a registry that allows multiple languages in its zone but does not trust the registrants to say the intended language of a registration.

A table should not have more than one entry for a particular base character. A table with more than one variant rule for the same base character requires that some names be evaluated by humans and will open the registration process to dispute. Such human intervention in the registration process may be unavoidable for some languages or for some registries, but it should be avoided if there is a desire for predictability in the registration process.

The description below does not specify where human intervention would happen because there are so many possibilities, based on the type of checking that a registry might want. It makes sense that a check might be made before step 1 or step 3 to be sure that the base registration meets any semantic rules for the zone, and that the intended language is in fact appropriate for the base registration. Some registries might also want another set of checks after step 5 to be sure that all the entries in the bundle are semantically appropriate for the zone. For example, if a zone prohibits mixed-script registrations, that check should be made both before step 1 (to check the base registration) and after step 5 (to check whether the variant-creation step created any mixed-script items in the bundle).

4.2 Description of CreateBundle

The CreateBundle process determines if a registration bundle can be created and, if so, fills that bundle only with valid labels.

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

The steps in the CreateBundle process are:

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

2) Perform the steps in ToASCII for the base registration. If ToASCII fails for the base registration, the system must stop and not register any of the label (that is, it must not register either the base registration or any created labels, even if those labels would have passed ToASCII). If ToASCII succeeds, add the result to the registration bundle.

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

   3a) Create the set of characters that consists of the candidate character and any variants.

   3b) For each character in the set from step 3a, duplicate the temporary bundle that resulted from the previous candidate character, and add the new character to the end of each partial label.

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

   4a) Process the label with ToASCII to see if ToASCII succeeds. If it does, put the label into the registration bundle. Otherwise, do not process this label from the temporary bundle any further; it will not go into the registration bundle.

5) The resulting registration bundle with the base registration and possibly other labels. Finish.

4.3 Example of expansion of a base registration into a bundle

[[Need at least one worked-through example of step 3 with interesting variant situations]]

5. Table format

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

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

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

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

The following is an example of how a table might look. The entries in this table are purposely silly and should not be used by any registry as the basis for choosing variants. For the example, assume that the registry:
- allows the FOR ALL character (U+2200) with no variants
- allows the COMPLEMENT character (U+2201) which has a single variant of LATIN CAPITAL LETTER C (U+0043)
- allows the PROPORTION character (U+2237) which has one variant which is the string COLON (U+003A) COLON (U+003A)
- allows the PARTIAL DIFFERENTIAL character (U+2202) which has two variants: LATIN SMALL LETTER D (U+0064) and GREEK SMALL LETTER DELTA (U+03B4)

The table would look like:

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

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

6. Examples

The following shows examples of the first two of the registry's options as described in Section 2.4 (putting all labels in the zone; putting only the base registration in the zone and blocking the rest). The third option (resolving some labels but blocking others) is an extension of these two and is not shown here.

The examples assume that the registry for the zone example.com uses the following very short table, which says that LATIN SMALL LETTER L (U+006C) has a single variant, DIGIT ONE (U+0031).

```
U+006C|U+0031
```

A registrant approaches the zone and requests a registration for the name pale.example.com, for which there are two name servers (x.example.com and y.example.com). After processing the base registration "pale", the registration bundle contains "pale" and "pa1e".

6.1 Example 1: allocating multiple labels

Assume that the registry for the zone example.com uses option 1 (allocating multiple labels) as its registration policy.

The registry allocates pale.example.com and pa1e.example.com to the registrant. The registry also creates a link in its registration database from pa1e.example.com to pale.example.com so that any changes to either the non-zone information or the zone information for one name will be reflected in the other name.

The registry adds the following four records to the example.com zone:

```
  $ORIGIN example.com.
  pale IN NS x.example.com.
  pale IN NS y.example.com.
  pa1e IN NS x.example.com.
  pa1e IN NS y.example.com.
```

Note that the registry might instead use DNAME records for allocating labels. If the registry uses DNAMEs, the registry would instead add the following three records to the example.com zone:

```
  $ORIGIN example.com.
  pale IN NS x.example.com.
  pale IN NS y.example.com.
  pa1e IN DNAME pale.example.com.
```

An end user who requests the name server for pa1e.example.com will get a positive response with the correct information.

6.2 Example 2: blocking labels

Assume that the registry for the zone example.com uses option 2 (blocking labels) as its registration policy.

The registry allocates pale.example.com to the registrant and blocks pa1e.example.com from being registered by anybody. The registry also creates a link in its registration database from pa1e.example.com to pale.example.com so that any changes to the non-zone information for pale.example.com will be reflected in the blocked name.

The registry adds the following two records to the example.com zone:

```
  $ORIGIN example.com.
  pale IN NS x.example.com.
  pale IN NS y.example.com.
```

An end user who requests the name server for pa1e.example.com will get a response of "no such name".

7. Owner implications of multiple labels

The creation of a registration bundle for equivalent or near-equivalent labels in a zone at the time of registration leads to many delegations. This leads to records in parallel zones which must be synchronized. That is, the owner of a registration bundle must keep the same information in the zone for each label in the bundle.

Using the examples from Section 6, assume that the owner of the label "pale" and "pa1e" creates a subdomain, "www". If the owner of "example.com" used multiple delegations for the labels, the owner of "pale" and "pa1e" would use two records:

```
  $ORIGIN pale.example.com.
  www IN A 1.2.3.4

  $ORIGIN pa1e.example.com.
  www IN A 1.2.3.4
```

An alternative for these two records, which helps the registrant keep their names in synch, would be:

```
  $ORIGIN pale.example.com.
  www IN A 1.2.3.4

  $ORIGIN pa1e.example.com.
  www IN CNAME www.pale.example.com.
```

If the owner of "example.com" used a DNAME record to make "pale" and "pa1e" equivalent, the owner of "pale" and "pa1e" could instead use one record:

```
  $ORIGIN pale.example.com.
  www IN A 1.2.3.4
```

8. Security considerations

Apart from considerations listed in the IDNA specification, this document explicitly talks about rules that a registry can define as part of the policy which can be applied in a zone. A registry can apply an variants table which solves some problems with homographs already outlined in the security consideration section of IDNA. This might be considered good for security because it will reduce the possible confusion for the user, and lower the risk that the user will "connect" to a service which was not intended.

Poorly-designed tables can cause security problems. For example, if a table creates variants for base characters that are not really variants, and names using those incorrect variants are allocated in the zone, an unsuspecting end user will get unexpected positive answers to DNS queries. For example, if the base character "a" has a variant of "e", and those variants are allocated in the zone, a user who looks up "bed.example.com" will get the information for "bad.example.com", even though the user could easily see that these two labels are different.

Users will likely expect that variants will be the same in zones and may make assumptions based on those expectations. However, the variants and how they are used will probably be different between zones, even for two zones that use the same language. This could lead to spoofing users by registering names that leverage the differences in rules between the zones.

9. References

9.1 Normative References

[IDNA] Faltstrom, P., Hoffman, P. and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

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

[UNICODE]   The Unicode Consortium. The Unicode Standard, Version 3.2.0 is defined by The Unicode Standard, Version 3.0 (Reading, MA, Addison-Wesley, 2000. ISBN 0-201-61633-5), as amended by the Unicode Standard Annex #27: Unicode 3.1 (http://www.unicode.org/reports/tr27/) and by the Unicode Standard Annex #28: Unicode 3.2 (http://www.unicode.org/reports/tr28/).

9.2 Non-normative References

[ICANN-IDN] ICANN, "Deployment of Internationalized Domain Names",
<http://www.icann.org/announcements/announcement-20jun03.htm>

[IDN-CJK] "Internationalized Domain Names Registration and
Administration Guideline for Chinese, Japanese, and Korean",
draft-jseng-idn-admin, work in progress.

10. IANA considerations

There are no IANA considerations for this document. The tables described in this document can be created by anyone. Tables at IANA are often considered to be authoritative, but languages have no one who is authoritative for them. It is unclear what value, if any, there is for someone to know what table a particular zone says it is using for registration. Further, the tables are expected to be updated at irregular times as new characters are added to the list of acceptable characters. Therefore, it is probably unwise for IANA to keep a registry of these tables.

11. Author's address

Paul Hoffman
Internet Mail Consortium and VPN Consortium
127 Segre Place
Santa Cruz, CA 95060 USA
phoffman@imc.org

A. Changes from the -01 document

Changed the intended status to Experimental

Added comments to the table syntax in section 5