INTERNET-DRAFT
June 19, 2002
Expires December 2002

John C. Klensin

Internationalizing the DNS -- A New Class
draft-klensin-i18n-newclass-02.txt

0. Abstract

Several mechanisms have been proposed for placing multilingual names (more properly, names normally written in non-ASCII character sets) into the DNS or addressing the need for multilingual access to the Internet in other ways. Most of them involve, to one extent or another, workarounds to the current system. This document proposes a "go back and fix it" approach, replacing the "IN" Class in the DNS with one that is not limited to ASCII from its initial definitions. Some of the deployment issues, politics, and other drawbacks are also briefly discussed.

The draft is being republished at this time, with a few corrections and some new text, because it is complementary to a recent draft by Ted Hardie [Hardie-Class] and because it may be part of the foundation for a different DNS internationalization proposal. The author is no longer convinced, if he ever was, that this proposal is adequate for DNS internationalization, at least without considerable enhancement.

A mailing list has been initiated for discussion of this draft, its successors, and closely-related issues in the Internationalization context at ietf-i18n-dns-newclass@imc.org. To subscribe to the mailing list, send a message to ietf-i18n-dns-newclass-request@imc.org with the single word "subscribe" (without the quotes) in the body of the message. To unsubscribe from the list, use that same address with the single word "unsubscribe" in the body of the message. Issues related to the relationship of the model proposed here to general issues of multilingual access to the DNS should be raised in the IETF IDN WG working group, see http://www.ietf.org/html.charters/idn-charter.html.

Table of Contents

1. Introduction and Context

There have been a large number of proposals, both inside and outside the IETF, for getting multilingual (or "internationalized") access to the DNS. With the exception of a few proposals that focus on doing the work in a separate system that permits searching (see [Klen-role, Klen-search, Meal-SLS], and others) and several suggested or deployed commercial products), all involve inserting the names into the existing DNS structure. This would be done either by using special conventions and preprocessing of international characters into an ASCII representation or by using a character set repertorie that includes more than ASCII [ASCII] and some encoding to place characters from that repertorie into the system.

To a considerable degree, while the objective is multilingual access to names (i.e., access from multiple languages), very few of the proposals address languages at all. For a number of good reasons, which are adequately discussed elsewhere, discussion has focused on access to the system with characters other than those that appear in ASCII and, in particular, characters drawn from ISO 10646 [IS10646]. This document, too, addresses characters, not languages, and "non-ASCII", "international", "multinational", and, following ISO's example, "universal character set" ("UCS") terminology are used interchangably below. When "multilingual" is used, it refers to the languages in which words or names appear, not what is placed into the DNS.

When the DNS was designed, it was anticipated that there might be future extension through the use of "Classes". All common current uses on the public Internet use the "IN" class. Two additional classes are known to have been defined and used: one for the old Chaosnet protocols and one for Hesiod-based protocols. Potential extensions via the Class mechanisms may have been one of the reasons that DNS labels and other fields were defined as binary, rather than ASCII, forms.

Applications using the IN Class have historically assumed labels limited to seven-bit ASCII characters and, more specifically, a "protocol element" character set derived from ARPANET "hostname" rules (see [Klen-3071]). This document explores the question of what the DNS, and "DNS names", might have looked like had we been designing it today. I.e., it assumes that multilingual usage would be a priority but that current technology and existing standards are available. It also outlines the issues associated with a transition mechanism.

The proposal is radical in the sense that it implies a major restructuring of DNS usage and, indeed, of the Internet, to make the DNS seamlessly capable of working with multinational (or, more properly, "universal") character sets. Such a restructuring is, and should be, quite frightening. It is worth considering only if the long-term risks and problems of other proposals are severe enough to justify a radical approach. It is the working hypothesis of this document that they are. At a relatively technical level, this would require changing every DNS resolver and server, and application that accesses either, on the Internet that wished to use non-ASCII names. Legacy (unconverted) systems would be at a significant disadvantage in referencing new names (some of which might use only a subset of ASCII characters but might still not be registered in the older Class), just as legacy systems were during the transition between Hostnames and the DNS. There are also a number of problems, such as the weaknesses of the DNS as a directory system, which it does not solve (see section 3.6, below).

It also does not significantly address the very complex issues of normalization, mapping, and canonicalization that are needed for the use of Unicode (or, in all likelihood, with any future alternate approach to a universal character set). The key issues in that area are addressed in the existing "stringprep" [stringprep] proposal and its profiles. See section 3.1 for further discussion on this issue.

2. Overview of the Proposal

Suppose we introduce a new Class (let's call it UC for "universal characters" just as a placeholder, but I hope that isn't what we would choose), which is just like IN (i.e., inherits its RR definitions, but see below), except:

* Labels and all fields containing text are defined as IS 10646 characters, coded in UTF-8 (or, in principle, some other _single_ system, i.e., we do not permit multiple character sets in the structure).

* A new RR is introduced that maps a new-type label (in the new class) into a restricted-ASCII (traditional) label. The intent is that the resolver then looks up the restricted-ASCII label in the IN class and proceeds as usual. Probably it would need to have "nothing else there" restrictions like CNAME, but (to put it mildly) I haven't thought that through. An alternative would be to adopt a search rule strategy, looking first with Qclass=UC and then Qclass=IN. The new RR might not be needed if one adopted a search rule strategy, or strong administrative rules requiring that all Class IN records be transposed into Class UC (see below).

* A second new RR is introduced that indicates that a delegated subdomain of a zone in Class=UC is in Class=IN, and not in Class=UC. This would be a variation on "NS", but would specify both the Class and the name associated with the relevant name server. (Note that a references from Class=IN to Class=UC makes no sense and mechanisms for it should not be provided.)

It is not clear at this time whether both the second and third "crossreference" RRs would be necessary, but almost certainly at least one would be.

This brief outline obviously leaves out many critical details which would need to be worked out, only some of which are explored in this draft of the document.

3. Technical alternatives and the deployment and transition nightmare

A "new class" proposal would obviously not be easy to deploy, but, realistically, neither are any of the other ideas if the definition of deployment involves users having access to Internet names drawn from a broad range of languages. It would cleanly separate "international character set" name spaces from the "ASCII" one --

i.e., "old" clients and systems would never see the non-ASCII types. In the international name space, English, and the character code points used to represent it, becomes just one of many such languages and their corresponding character code points. It might even let us fix a few other things along the line, as long as they were sufficiently straightforward to not create significant delays. E.g., there are several RR types in the current Class that are either obsolete or have never been widely used, and we might be able to eliminate them by not carrying them forward.

While other transition models are possible, the cleanest one would be to conclude that the new Class was intended, over time, to simply obsolete and replace Class=IN. If registrations in Class=IN were transferred into (or explicitly referenced from) the new class (or a "search rule" system was employed), then the transition model would be very similar to that of the hosttable-> DNS transition. In particular, "old" clients and systems would see a smaller and smaller fraction of the Internet until they converted and we would expect some user-level tools to arise to work around slow conversions.

3.1 Preparation and comparison of names

Subsets of ASCII, or character codes whose character repertoires are themselves subsets of ASCII, have long been the character repertoire of choice for the protocol elements of protocols that use characters in such elements. While some of the reasons for this --arguably including the decision to use characters from a Roman- (Latin-) based alphabet-- are simply historical, ASCII has the advantages of containing a very small (by world averages) set of characters, of permitting an extremely easy case-mapping algorithm (and case-mapping is important in Roman-based and several other languages), of requiring no "composed" characters, and of raising no significant issues with canonicalization, collation, or identity-matching.

To varying degrees, as soon as the character repertoire moves beyond the requirements of ASCII, comparison issues intrude: it is necessary for a DNS server to determine whether the name specified in a query matches the name that appears in its tables for a domain. And that, in turn, requires either that strict rules be applied to how names are stored and how queries are presented or that the server be able to interpret a somewhat-ambiguous (or "fuzzy") query. The latter option is infeasible given the design of DNS servers (although non-DNS systems might permit it -- see section 3.6 and [klen-role]). The former has been the focus on the "nameprep" efforts within the IDN Working Group.

In general, the mechanisms and rules being developed as part of the "nameprep" effort would need to be applied to a "new class" system, just as they would need to be applied to "edns/UTF-8" or "ACE" systems. However, one potential advantage of a "new class" approach, and possibly of an EDNS-based one, is that it would be possible to move some of the comparison and mapping functions out of a pre-query sequence on the client and onto the server. That, in turn, would permit case mapping and matching for non-ASCII texts to be handled in a way much more similar to ASCII text today (e.g., storage of mixed case materials in zone files with queries in any case and return of the registrant-desired form). To the extent that centralizing complex functions on the servers improves reliability, it would also have that benefit.

3.2 Registrations in both places

A "multilingual" name registrant could choose whether to register the multilingual name exclusively or whether to register ASCII-based names as well (giving most of the useful properties of the two-sided business card analogies). Such ASCII-based names could be registered in the new class; they could presumably also be registered in the old class for compatibility with legacy systems. We would not expect reverse mappings to work from IN-space to UC-space; PTR lookups in Class IN would yield ASCII names; PTR lookups in Class UC would yield IS 10646-based names. And we would expect all other fields that contain text to contain IS 10646-based text, not just ASCII. Finally, moving critical portions of the normalization and comparison work to the server might make it easier to deal with some of the issues of in superimposing internationalized names on DNNSec a bit easier, or at least possible to have better-behaved.

There is probably no practical way to automate dual registrations in the general case (keeping in mind that naming and identification issues that exist near the root also exist deep in the DNS tree), so decisions about legacy registrations would need to be administrative and policy based. See section 5. Search rules (see next section) might be an acceptable substitute for dual registrations, but have their own disadvantages.

3.3 Search rules and search failures

Unless all relevant records in Class=IN are copied into Class=UC and the two are kept synchronized (very difficult if not impossible to maintain), there will be a requirement for searching from the newer class to the older one. That requirement could, in principle, be kept in the servers and off the network by providing for new servers to automaticaly search in Class=IN if nothing is found in Class=UC without intervening interactions with the resolver. This could introduce significant complexity and a number of special cases into the server and might or might not be wise. Since a new Class causes multiplicative effects on the number of probes potentially required to complete a search, minimizing the number of similar RR types in the new Class becomes technically advantageous as well as aesthetically so (see section 4).

The potential need to make queries in both Class=UC and Class=IN for a given user-supplied name provides an immediate, and strong, reason why the fundamental domain hierarchy structure of both Classes should be identical, even if the servers are not. If identity of servers is not practical (it probably would not be significantly below the top level, if there), the portion of the Class=UC tree that shared names with the Class=IN tree would need to be identically structured. Almost by definition, as non-ASCII non-terminal nodes are introduced into the Class=UC tree, that tree would diverge from, or become a superset of, the Class=IN one. The alternatives would, at best, be hopelessly confusing to users.

But, if searching mechanisms from Class=UC to Class=IN will be in regular use, it is tempting to rely on those mechanisms rather than doing any forward copying of data. This would increase overhead in comparison to having all information copied into the UC class as early as possible, but the range of alternatives needs to be studied carefully, especially with regard to domain trees that contain non-ASCII names and UC-capable servers at some nodes and only ASCII names and legacy servers at others. The special, cross-Class NS RR suggested above would help with such trees, but some searching strategies might make strict bottom-to-top conversion of subtrees (rather than level-skipping) very valuable if not necessary.

Having two classes also raises issues for which the answers seem obvious, but decisions must be made and made explicit. For example, it seems clear that one should search

```
   ((QClass=UC, Qtype=MX, ...)
    (QClass=UC, Qtype=A6, ...)
      ...
    (QClass=IN, Qtype=MX, ...)
      ...
```

But, at least in theory, a case could be made for looking for MX RRs in "IN" before looking for address records in "UC".

3.4 A "new class" solution versus an "edns/utf-8" one.

Some of the proposals before the IDN working group (and elsewhere) depend on the use of "extensible DNS" ("edns") facilities to permit extended labels and the use of UTF-8 encoding in them. Proponents point out that edns is extremely useful for IPv6 and DNNSEC, so will probably deploy quickly anyway; its use for non-ASCII DNS labels would both benefit from and reinforce those deployment pressures. One of the barriers to the deployment and heavy use of extensible DNS [RFC2671] is that its use in the current, Class=IN, environment depends somewhat on updating of intermediate servers. In other words, an updated client and updated primary server may not be able to properly interoperate because caches or secondary servers may still be running older code.
In principle, and probably in practice, this is not an issue with a new Class: absent serious errors of configuration, name servers delegations and caches for the new class would be only to servers supporting that

class.

3.5 A "new class" approach versus an "ACE" one.

Several of the proposals in the IDN working group (and elsewhere) depend on encoding ISO 10646 characters into an ASCII-compatible format (an ASCII-compatible encoding, hence "ACE") so that the names, however ugly, would survive passage into applications that have intrinsic seven-bit limitations. That group of applications is somewhat more diverse than what is usually thought of as "internet applications". For example, X.509 certificates are used in SSL and assume seven-bit characters. The ACE codings would work with those applications, although they would look nothing like the graphic characters of the original character set and language.

While this document has assumed using the UTF-8 encoding of IS 10646 directly in the names and labels of the UC class, UTF-8 is just another encoding and not an especially efficient one. There may be applications-based arguments for using an ASCII, or ASCII-compatible, encoding to represent character codes in the new Class as well. However, since all codes in the new Class would be using the same system, one could devise a system that did not require a switching or labeling mechanism to identify the use of the coding system versus the appearance of codes in the ASCII range intended to be interpreted as ASCII. I.e., prefixes or suffixes might become unnecessary and it might be possible to use higher-density encodings, such as MIME's base64, or a serious compression algorithm applied to UCS-2 or UCS-4 (or the similar UTF-16 or UTF-32), rather than UTF-8 or those encodings more commonly suggested as ACE mechanisms.

3.6 A "new class" approach versus a "directory" one.

Many of the issues raised in [Klen-role] are not addressed by this proposal. Neither it, nor any other DNS-based solution, would turn the DNS into a searchable directory ([klen-search] does provide a model for a searchable directory, but the work is not done in the DNS). Nor can they address imprecise matching, keyword matching, nearest applicable server location, searching on the content of data fields, and so on. This proposal does provide a plausible solution to reverse-mapping problems, deployment, and has known scaling properties: all areas where the notions outlined in [klen-role] are weaker. It would probably be somewhat faster than a directory approach layered on the DNS, since there would be no requirement for a two-stage lookup process. But, ultimately, the two proposals are complementary: There is a strong applications case for introducing a directory layer. While the directory layer could be used to support multilingual names -- treating the ASCII-based names in the DNS as protocol elements rather than names that ought to be user visible -- it could also be used with a DNS that actually and cleanly supported multilingual names as suggested here.

3.7 Another look at legacy applications

As suggested above, there are some applications, many with origins outside the IETF, that cannot be easily upgraded to use of non-ASCII (or, generally, non-seven-bit), character codes. It is difficult to know what to do about those applications. If we are really serious about converting the Internet to support applications in all languages (which is ultimately the assumption underlying this document), then the answer may be more clear: the overhead of dealing with the UCS to ASCII interface ought to fall on those applications, as an intermediate step until the protocols themselves can be upgraded. In other words, we might anticipate a four-stage conversion process for those applications:

(i) Completely legacy (non-updated) code would continue to reference Class=IN (no other option is possible).

(ii) Applications code would be upgraded to make QClass=UC inquiries and to represent the UCS codes for their databases and presentations in some ASCII-compatible form compatible with their protocol definitions. In other words, the DNS would return some native form variation on UCS, conversion to ASCII-compatible form would occur, when needed, on the applications side of the DNS interface.

(iii) The protocols would be upgraded to international norms and usage.

(iv) The applications code would be changed to conform to the new protocols, eliminating the workaround of stage (ii).

If these conversions and downgrades, espeically those of step (ii), are incorporated into the DNS, we are stuck with their overhead and appearance forever. That is, of course, also true of any encoding tricks (e.g., the "ACEs" under consideration in the IETF IDN WG, e.g., [DUDE]) used to represent non-ASCII names in Class=IN without putting applications at obvious risk. And different applications, with different constraints, may have to convert them to application-specific formats anyway. Somewhat different strategies, available with the new class approach only, could eliminate the need to make workarounds and kludges a permanent part of their systems and the Internet infrastructure.

3.8 New Classes and IPv6 Transition

The transition to IPv6 has raised an interesting general DNS issue because it may require that DNS servers support a dual stack arrangement to permit access of records from resolvers based on both IPv4 and IPv6 systems. (This document will not attempt to fully explain that problem and its implications.) It is possible that, were a new Class developed as an incremental replacement for Class=IN, it could be used as a mechanism for handling IPv6 queries.

4. Bringing RR types forward

In principle, one could populate the UC Class with all of the types of the IN one, possibly eliminating those that are clearly obsolete, as mentioned above. A narrow reading of many of the existing definitional documents might even require this, although we can be assured that no queries or registrations in class=UC exist today. But it might be interesting to evaluate the implications of taking a harder line, partially to shorten search paths and minimize the size of zone files. Several RR types have been added "experimentally"; it would probably be wise to leave those for which there isn't considerable deployment and justification behind. We might also consider leaving AAAA RRs behind as obsolete or redundant, since A6 is the more general of the two. And, more radically, one might consider eliminating type A RRs, writing IPv4 addresses in IPv6 form, e.g.,

```
   kakameymi.example.   UC  A6  0  ::FFFF:10.0.0.44
```

and letting APIs to resolvers translate them back into IPv4 format if needed. By doing this, the potential need to query for A6, AAAA, and A RRs in sequence would disappear, resulting in some performance improvement, especially for the "not found" cases.

Of course, similar logic might apply to a lighter-weight successor of A6, or even a variation of AAAA with a specialized high-order coding convention.

This might raise entry barriers too much to be worthwhile, but, if feasible (and we really believe in IPv6 deployment), it would yield a much cleaner environment for both forward and reverse mappings.

On the other hand, eliminating other types in favor of A6 might be advancing the technology in too many ways at once. For example, while the A6 RR appears to be fully general, there is so far little few real experience on using it (or other IPv6 RRs) and none of that experience is at large scale. It may be wise to go somewhat cautiously into directions that tie the new class to such less-than-complete-tested approaches.

5. Pushing into layers eight through ten

(For those who don't know, these layers have become an internal joke in the IETF community whose exact origins are unknown to this author. The layers are characterized as "financial", "political", and "religious", with some debate about the order.)

A new DNS class really would be new. Current Internet administrative procedures and lines of authority with regard to the DNS have assumed that Class=IN is the only class at issue. It is to be hoped that IETF could identify technical criteria and other important tradeoffs but leave the non-technical issues and resolution of policy and political tradeoffs to ICANN and related bodies.

5.1 The root server question

The design of the DNS is such that there is no inherent reason why root servers for a new class would need to be the same as those for IN. However, the same considerations for root server selection that apply to the Class=IN root [rfc2870] would presumably apply to the Class=UC root as well. There would be several other administrative and operational advantages for keeping many or all of the root servers the same --or at least co-locating them-- as long as loads, software availability, and similar factors permitted.

5.2 Other administrative challenges

Just as the root servers would not need to be the same, the introduction of a new Class would, in theory, permit revisiting the entire top-level structure and administration of the the Class=IN DNS (e.g., there would be no requirement that the TLD names or model be the same). Doing so would probably be unwise if we wanted to see this deployed in our lifetimes, but the possibility must be identified and, at least briefly, considered.

5.3 Thinking about deployment

It is clear that, like any other multilingual approach, software supporting a new DNS class would deploy much more quickly in areas which clearly need it than in areas that perceive they do not. The requirement for communication with, and access to sites in, non-English-speaking areas would tend to drive deployment in other areas with this and many other approaches. The so-called "ACE" approaches within Class=IN (and perhaps some others) using the IN Class would permit non-updated sites to see multilingual names in their ugly encoded forms; it is possible that would actually act as a disincentive to updating and conversion since the names would still be somewhat visible; this proposal would not make multilingual names available in any form to legacy systems. On the other hand, some have argued that the use of ACE-type systems in the current Class would make the full horrors of kludgy and insufficient implementation of multilingual names visible to all, encouraging the deployment of either the type of system outlined here, or a search-based one, or both.

Whatever thinking is done about deployment tradeoffs should consider Internet growth rates, especially in non-English-speaking areas. Whatever solution is adopted, we will need to live with it for a long time. If no old systems are ever converted, but new ones installed after a particular date have updated software installed, and "doubling every year" behavior continues, then the legacy base represents half of the Internet a year later, a quarter of the installed base a year after that, and so on. So, a sufficiently important change, incorporated into relevant shipping software, has a very large percentage impact even if there is no actual updating of systems. This is something to keep in mind, especially if the alternatives are overhead-laden kludges that we will need to support forever.

6. Summary

<<To be supplied in the next draft>>

7. References

7.1 Normative references

[ASCII] American National Standards Institute (formerly United States of America Standards Institute), X3.4, 1968, "USA Code for Information Interchange". ANSI X3.4-1968 has been replaced by newer versions with slight modifications, but the 1968 version remains definitive for the Internet.

[IS10646] ISO/IEC 10646-1:2000 Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane and ISO/IEC 10646-2:2001 Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 2: Supplementary Planes.

7.2 Non-normative references

[RFC2671] Extension Mechanisms for DNS (EDNS0). P. Vixie. August 1999.

[RFC2870] Root Name Server Operational Requirements. R. Bush, D. Karrenberg, M. Kosters, R. Plzak. June 2000

[hardie-class] Hardie, T., "A DNS RR for Pointers to RRs outside class IN", work in progress, (http://search.ietf.org/internet-drafts/draft-hardie-out-rr-00.txt)

[klen-3071] Klensin, J., "Reflections on the DNS, RFC 1591, and Categories of Domains", RFC 3071, February 2001.

[klen-role] Klensin, J., "Role of the Domain Name System", work in progress (http://search.ietf.org/internet-drafts/draft-klensin-dns-role-03.txt)

[klen-search] Klensin, J., "A Search-based access model for the DNS", work in progress (http://search.ietf.org/internet-drafts/draft-klensin-dns-search-03.txt)

[meal-sls] Mealling, M. and L. Daigle, "Service Lookup System (SLS)",

work in progress.
(http://search.ietf.org/internet-drafts/draft-mealling-sls-00.txt)

[stringprep] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ('stringprep')", work in progress, (http://search.ietf.org/internet-drafts/draft-hoffman-stringprep-03.txt)

8. Acknowledgements

Rob Austein and Randy Bush made very significant contributions to the thinking and some of the text that went into early versions of this draft through a series of email discussions. Others, including Marc Blanchet, Vint Cerf, Kilnam Chon (with apologies for writing his name in ASCII characters rather than Korean ones), Patrik Faltstrom (with apologies for the ASCII transposition), Paul Hoffman, David Lawrence, and Zita Wenzel have made suggestions or challenged some of the ideas in their embryonic form, leading to clarifications and clearer thinking. More recent discussions with Erik Nordmark, Ted Hardie, and Dan Oscarsson, and observation of many discussions on the IDN WG list contributed to the changes in draft 02. Patrik Faltstrom first pointed out the possible IPv6 implications of a new class approach. The author, of course, bears ultimate responsibility for the ideas as presented.

9. Author's address

John C Klensin
1770 Massachusetts Ave, #322
Cambridge, MA 02140
klensin+irnss@jck.com

Expires December 2002