VeriSign®

VeriSign® White Paper: A Proposal for DNAME Equivalence Mapping for TLD Strings

Where it all comes together.™

Abstract

RFC2672 (Non-Terminal DNS Name Redirection)[1], an IETF standards track document defines a new Domain Name System (DNS) Resource Record called "DNAME", which provides the capability to map an entire sub-tree of the DNS name space to another domain. This paper discusses the possibility of creating either transliterations or local language equivalents of Top Level Domains (TLD) as ASCII Compatible Encodings (ACE) and relying upon the DNAME construct for mapping such namespaces directly onto existing generic and country-code TLDs. The purpose of this paper is to examine this approach technically and to outline benefits and challenges to such an approach.

Table of Contents

[1] RFC 2672 (Non-Terminal DNS Name Redirection, M. Crawford, August 1999, ftp://ftp.isi.edu/in-notes/rfc2672.txt

I. Introduction

The Internet is a global communication medium that has been a phenomenal catalyst for trans-national and cross-language information exchange and interaction. The origins of the Internet as a government-funded research network center on US and western European organizations and the cooperative development of technical standards. The Internet Engineering Task Force (IETF) has long been the coordination point for authoring the basic building blocks, protocols, which are the rulebooks which define how Internet applications and services interact. The fundamental basis of the Internet's protocol suite, American Standard Code for Information Exchange (ASCII), is a character set which reflects the scripts used by its major contributors. While ASCII can broadly represent English and Western European written languages, it is insufficient for most of the world. The goal of this paper is to provide an overview of the existing ASCII-based DNS, to summarize work establishing RFCs for Internationalized Domain Names (IDN), and to examine the potential application of an existing Internet standard to support the demand for truly internationalized DNS.

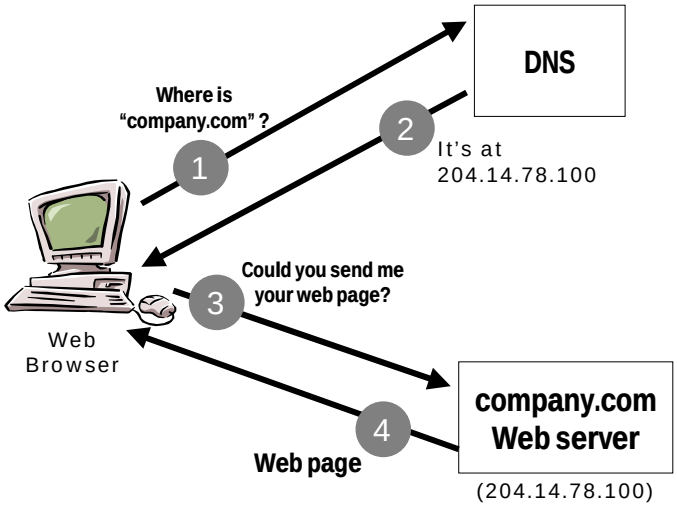
RFC2672 (Non-Terminal DNS Name Redirection), an IETF standards track document defines a new DNS Resource Record called "DNAME", which provides the capability to map an entire sub-tree of the DNS name space to another domain. It differs from the CNAME record which aliases a single node of the name space. In this fashion and in concert with RFC 3492 (Punycode: A Bytestring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)) which establishes an approach for representing Unicode characters as ASCII strings within the DNS, this paper discusses the possibility of creating either transliterations or local language equivalents of Top Level Domains (TLD) as ASCII Compatible Encoding (ACE) and relying upon the DNAME construct for mapping such namespaces directly onto existing generic and country-code TLD's.

The purpose of this paper is to examine this approach technically and to outline benefits and challenges to such an approach. Because we assume the technical background of readers of this work will vary, the following sections describe the DNS and work towards internationalization in basic terms. Subsequent sections include technical details regarding the specific proposal as well as illustrations of potential policy issues. While the content in some cases is fairly technical, the level of detail is a necessary context for understanding the application proposed.

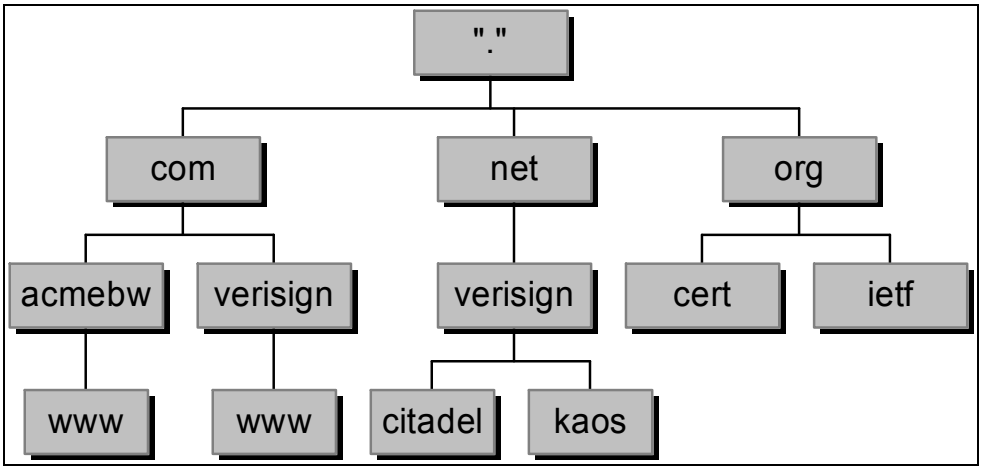
II. Domain Name System (DNS) Overview

The *Domain Name System* is a distributed database system that functions as the Internet's naming service, mapping human-readable *domain names* into IP addresses, mail routing information, and more. DNS allows people to use names (e.g., *"company.com"*) to identify Web servers, rather than IP addresses. DNS performs the

translation between the name and the IP address or addresses. When an Internet user types *"company.com"* into a Web browser, DNS translates that domain name into an IP address. The browser then connects to the Web server at that address. The diagram below demonstrates this simple process.



Domain names are actually paths in an inverted tree of *nodes*, called the *name space*. The name space has a structure like that shown in the figure below.



Each node has a textual label, from zero to 63 bytes in length. The domain name of a node is simply the concatenation of labels on the path from that node to the root of the inverted tree, with dots separating the labels.

These domain names serve as indices into the distributed database. Within the database, data is stored as a triple of domain name, *class*, and *type*, where class indicates the protocols used on the network described by this datum and type indicates the function of the datum.

The distributed database itself is partitioned into zones, which are groups of contiguous nodes in the name space. Each zone contains all of the data indexed by the domain names of nodes in the zone. One or more *name servers* store each zone. A name server with complete information about a zone is said to be *authoritative* for that zone.

To retrieve arbitrary data in the distributed database, name servers use a system called *name resolution.* In the absence of more specific knowledge about a domain name, a name server begins its search at the root of the name space (i.e., by querying name servers authoritative for the root zone, also called a *root name server*). The name server presents the root name server with the domain name of the node for which data is sought. The root name server, if it does not know the final answer, refers the name server to the name servers authoritative for the closest subordinate zone it knows about. For example, a root name server might refer a querier interested in www.verisign.com to the name servers authoritative for com, a known ancestor of the node the querier seeks. This process continues until the name server performing name resolution finds the data it seeks.

Some nodes in the name space are actually aliases for other nodes. If a name server encounters such an alias during name resolution, it must restart the name resolution process, replacing the domain name of the node it sought with the domain name the alias refers to. For example, a name server looking up the address of www.verisign.com might find (from the name servers authoritative for the zone that contains www.verisign.com) that www.verisign.com is an alias for verisign.com. It would then restart name resolution, looking for the address of verisign.com.

Since aliases cause this redirection from one node to another, nodes that are aliases cannot index any other data. Allowing a node to act as an alias while simultaneously indexing, say, an address would lead to ambiguity: If a name server looked up the address of the alias, would it find the address attached to the alias node or an address attached to the target of the alias?

Aliases are also restricted to nodes of the name space without child nodes. If an alias node were allowed to have children, this would also give rise to an ambiguity: While resolving the domain name of one of the child notes, a name server might encounter the

alias. Should it restart the query at the target of the alias and descend the name space from there, or should it continue descending through the alias, ignoring it?

III. Internationalized Domain Names (IDN) overview

Background

The core DNS RFCs[2] allow domain names to comprise an arbitrary sequence of bytes or *octets*. However, these RFCs also recommend that domain name syntax be limited in practice to avoid causing problems with legacy applications at the time DNS was deployed. RFCs dealing with electronic mail also specify the syntax of domain names. The recommended syntax is letters of the Latin alphabet, European digits (0-9) and the hyphen, all as represented in the US-ASCII character set. Since then, many applications have codified these suggestions into near requirements. Applications and even DNS servers and resolvers expect domain names to follow this syntax. The result today is that DNS domain names are effectively restricted to a very small set of characters.

The Internationalized Domain Name (IDN) Working Group[3] of the IETF was formed in the first half of 2000 with the goal of developing standards to internationalize domain names. The group's charter was to specify a set of requirements and an IETF standards-track protocol(s) to allow a broader range of characters to be used in domain names. In March of 2003, the group completed its work with publishing of the IDNA RFCs (3490, 3491, 3492 and 3454), Domain names such as 記念.*com* are as valid as *verisign.com*.

Character Encoding Terminology

It's important to understand the terminology relating to representing characters in information processing. An ***abstract character repertoire*** is a set of characters, called *code points*, from one or more alphabets or *scripts*. (A script is the set of letters used to write a particular language.) For example, an abstract character repertoire could include the uppercase and lowercase letters of the Latin alphabet, European digits and various punctuation. A ***coded character set*** assigns a number (a non-negative integer) to each character in an abstract character repertoire. In other words, it gives an index

[2] Principally RFC 1034 and RFC 1035.
[3] See *http://www.ietf.org/html.charters/idn-charter.html* and *http://www.i-d-n.net*.

value to each character in the repertoire. A ***character encoding form*** maps the set of integers used in a coded character set to *code units*. A code unit is an integer occupying a specified binary width in a computer architecture, such as an 8-bit byte. The encoding form enables characters to be represented as actual data in a computer. Finally, a ***character encoding scheme*** is a mapping of code units into serialized byte sequences. Character encoding schemes are relevant to the issue of cross-platform persistent data involving code units wider than a byte, where byte-swapping may be required to put data into the byte polarity canonical for a particular platform.

The mapping from a set of characters to a serialized sequence of bytes that can be used to store or transmit data is called a ***character map.*** A character map includes an abstract character repertoire, a coded character set, a character encoding format and a character encoding scheme. Many people use the general and overloaded term *character set* instead of the more precise term character map.

The Unicode Standard

There is an alphabet soup of character sets in use throughout the world. Each of the world's major scripts, from the Latin alphabet used by Western languages to the complex scripts used to write Asian languages, have many different character sets. This profusion of different codes to exchange information has led to the development of a universal character set called Unicode.[4] The current version of the Unicode standard, version 4.1, contains 97,655 distinct coded characters. These characters cover the principal written languages and symbol systems of the world. Since Unicode code points are numbered in a 20-bit space, theoretically just over one million code points can be represented.

There is universal agreement in the IDN Working Group that standards for internationalized domain names should be based on Unicode. In other words, IDNs will be composed of a sequence of Unicode characters or code points. The ongoing discussion has focused on how to represent these IDNs for use in the DNS protocol.

Encoding Unicode Characters

[4] Unicode is related to an international standard called ISO/IEC 10646. Both Unicode and ISO/IEC 10646 are based on the same abstract character repertoire and coded character set and the two standards track each other. While a detailed description of differences is beyond the scope of this paper, the important point is that both standards contain the same characters and will in the future.

A sequence of Unicode code points can be represented in multiple ways by using different character encoding schemes (CES). Two popular CESs are UTF-8 and UTF-16.[5]

UTF-8 is a variable-length encoding, which means that different code points require different numbers of bytes. The larger a code point's index number, the more bytes required to represent it using UTF-8. For example, the first 127 Unicode code points, which correspond exactly to the values used by the US-ASCII character set (which maps only 127 characters), can be represented using only one byte in UTF-8. In fact, their representation in UTF-8 is the same as in US-ASCII. UTF-8 can require up to four bytes to encode certain code points, however. A reasonable criticism of UTF-8 is that it penalizes certain scripts by requiring more bytes to represent those scripts' code points. The IETF has made the UTF-8 its preferred default character encoding for internationalization of Internet application protocols.[6]

UTF-16, on the other hand, is a fixed-width encoding: all Unicode code points can be represented with a two-byte value. (The full 20-bit range can be indexed with only 16-bits because a reserved range in the first 16-bits, called the *surrogates area*, is used to encodes values beyond 2^{16}.)

Both UTF-8 and UTF-16 require an "8-bit clean" storage and transmission medium. Unlike US-ASCII, which only uses seven bits per byte to encode characters, UTF-8 and UTF-16 need all eight bits per byte. (Such 8-bit data is informally called *binary data*, to contrast it US-ASCII data.) Since historically domain names have been able to be represented with 7-bit US-ASCII characters, not all applications that process domain names preserve the status of the eighth bit in each byte; in other words, they are not 8-bit clean. This issue led to significant debate in the IDN Working Group and influenced the direction of the standards development.

Progression of the IDN Working Group

The members of the IDN Working Group considered the 8-bit clean issue from the very beginning. Since it would seem that internationalizing domain names would invariably involve storing and transmitting binary data (i.e., Unicode strings encoded in UTF-8, UTF-16 or another encoding), and since applications that process domain names do not expect such data, it would seem that the internationalization process faced an early impasse.

[5] UTF stands for Unicode Transformation Format.
[6] See RFC 2277.

ASCII Compatible Encoding

Very early the working group considered the approach of using an *ASCII compatible encoding*, or ACE, to encode the Unicode strings that would make up IDNs. This technique represented binary data using only US-ASCII characters. The concept is very similar to the Base64 encoding used by the MIME email standards.[7] But whereas Base64 uses 64 characters from US-ASCII, including uppercase and lowercase, the ACE approach requires a smaller subset of US-ASCII, since DNS does not preserve the case of letters.

Various ACE algorithms had been proposed and published as Internet-Drafts. All the algorithms attempted to compress the input string before representing the output in some US-ASCII subset. Different algorithms had different compression goals (and yields) and encoded data using slightly different subsets of US-ASCII.

One feature of all the ACE algorithms proposed is that they applied to individual labels in a domain name. Thus it's possible that one label of a domain name could be internationalized and another not. Consider the example IDN given early in this paper, 記念.*com*. The first label is internationalized and would need to be encoded with an ACE, but the second label is the familiar *com* and can remain unchanged.

Another feature common to all the ACE proposals was some kind of tag to indicate that a particular label is, in fact, ACE encoded. Most proposals specified a prefix, though at least one specified a suffix.

One of the early ACE algorithms proposed was RACE (Row-based ASCII Compatible Encoding), and was widely implemented as a result of its use in VeriSign's Internationalized Domain Names Testbed. To give an example, the RACE encoding of the domain name 記念.*com* in the early testbed was `bq--3cfbqx7v.com`. The two Chinese characters (記念) encode to `3cfbqx7v`, and `bq--` is the prefix indicating that particular label is encoded in RACE.

Since then RFC 3492 has been published defining the ACE algorithm known as punycode. Punycode was chosen for its compressibility as well as its ease of implementation. Thus the testbed domain name of記念.*com(*bq--3cfbqx7v.com) was migrated to the punycode of xn--h7tw15g.com.

IDN in Applications (IDNA)

[7] See RFC 1521.

While an ACE allows an IDN to be represented in a form that can be handled by the currently deployed DNS infrastructure on the Internet, an ACE algorithm alone is not a full solution: a standard for IDNs would need to explain further details, such as where the ACE encoding and decoding should be performed.

Several comprehensive IDN solutions using an ACE as a component were proposed in the working group, but the final RFC was published as 3490 *Internationalizing Domain Names in Applications (IDNA)*.[8] As the name suggests, this draft proposes that the ACE encoding and decoding be performed solely in applications that use IDNs, such as browsers, mail user agents and other network clients (Telnet, FTP, etc.). The RFC calls for the creation of a *presentation layer* in IDN-aware applications that would be responsible for the ACE encoding and decoding. This new layer in the applications' architecture would be responsible for encoding any internationalized input in domain names into ACE format before the corresponding domain name is sent over the network (i.e., before resolution is attempted). Likewise, this new layer is responsible for decoding the ACE format in IDNs and rendering the appropriate internationalized characters for the user.

The beauty of this solution is that all deployed DNS infrastructure, from resolvers to name servers, would work without modification. Critics argued that the solution is a "hack" and aesthetically offensive. Admittedly, any ACE-based solution is designed to work in the short term: because the installed base of DNS software cannot reliably accommodate binary data, a solution using ACE would appear to be the only solution that can be deployed quickly. Any IDN solution requiring transmission of binary data would require upgrading existing resolvers and name servers. The IDNA draft makes the assumption that it's easier to upgrade user applications than the underlying DNS infrastructure.

Name Preparation

The US-ASCII character set is simple enough that the syntax for domain names has been very simple: only alphanumeric characters and the hyphen character are allowed, and any comparisons need to be case-insensitive. Unicode is considerably more complicated, with multiple ways to represent the same string. Consider, for example, Latin letters with diacritical marks, such as **Ä**. This letter can be represented with a Unicode code point called LATIN CAPITAL LETTER A WITH DIAERESIS. However, another valid representation of this character is the code point LATIN CAPITAL LETTER A followed by the separate code point COMBINING DIAERESIS. Through a process called *normalization*, these two code points can be combined into the single code point representing the composed character. In fact, depending on the type or *form*,

[8] This draft was formerly entitled *Internationalizing Host Names in Applications* and the IDNA abbreviation has remained.

normalization can also reverse the process and decompose combined characters. Without normalization, Unicode strings cannot be reliably compared for equality.

A further issue is characters that would cause confusion in IDNs and should be prohibited. For example, Unicode defines several different kinds of space characters. The consensus of the Working Group is that characters such as these are too confusing in domain names—users could not be expected to understand and input them—and should be prohibited.

Another of the IDNA RFCs is RFC 3491 *Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)* (often referred to as simply "nameprep"). This draft specifies operations that must be performed on IDNs before they can be used—compared with other IDNs, send to a name server for resolution, etc. The RFC references four operations all of which are defined in RFC 3454 *Preparation of Internationalized Strings ("stringprep")*:

1. Mapping: Each character in an IDN is compared against a mapping table, which allows certain characters to be replaced with other characters. For example, the most basic kind of mapping performed in this step is *case folding*, or converting all characters to lower case.

2. Normalize: The IDN is normalized according to form KC, a process defined by the Unicode Consortium.[9] This process composes characters as described above and performs additional transformations on the domain name.

3. Check for prohibited characters: If the resulting IDN contains any prohibited characters, such as the spaces mentioned above, the domain name is illegal and must be rejected.

4. Check for bi-directional characters: If the string contains characters from scripts that are both right to left and left to right, the domain may be illegal if it does not follow the rules in RFC 3454 and may have to be rejected.

IV. DNAME proposal

As outlined above, the DNS can accommodate domain names internationalized at the at the top level label by assigning DNAME resource records for each language equivalent of the TLD. This paper proposes that TLD registry operators establish local language representations and their ACE form for each language community in which they

[9] See UTF-15, *http://www.unicode.org/unicode/reports/tr15*

provide registration services and to coordinate with ICANN to place the necessary DNAME records in the root zone.

In order to better illustrate this proposal, a series of appendices are included. Appendix 1 contains examples of ACE mappings for local language representations of company and network in a sample of languages. Appendix 2 depicts the precise information that would appear in the root zone. Appendix 3 provides a hierarchical view of the zone structure. Appendix 4 provides a decomposition of the resulting query for ML.ML-TLD domains and their mapping back onto the ASCII namespace.

Leveraging the utility of a DNAME-based approach to internationalizing the top-level portion of a domain name provides a number of benefits.

First, it ensures a direct relationship between existing ASCII TLDs and local language representations within the existing ICANN framework. In this fashion, the namespace is not expanded in the traditional sense though resolution is provided with an anchor back to the original ASCII TLD. As such, the existing ICANN agreements with TLD operators could continue to be the vehicle used to manage registry operations.

Secondly, the potential for end-user confusion is lessened. By coordinating the representations of TLDs in a language a registry offers services and mapping it back to the ASCII form, consumers worldwide are provided a local language construct mapped onto the ASCII DNS structure, a format well understood.

Thirdly, this approach defines a logical methodology for leveraging existing resources without the need to hastily create new internationalized TLDs. As evidenced by the extensive and time-consuming process for the latest round of new TLD registries, under this approach, ICANN need only coordinate with existing registries to define the appropriate mappings by language.

And finally, because the DNAME mapping “unifies” the local language versions of a TLD into a single namespace, end-users need not take any action. By registering a domain name of the form ML.TLD, the consumer has automatic access to the form ML.ML-TLD where ML-TLD represents each language DNAME record created. Of note, this lessens the opportunity for UDRP disputes because no matter how many DNAME mappings are created for a TLD, there remains a single registrant of the second-level name.

The remainder of this paper discusses technical and policy issues which may impact the implementation of the technique described herein.

V. Technical issues, dependencies, assumptions

Location of DNAME record

A DNAME record to allow equivalence mapping of a new TLD to an existing TLD would have the following form:

```
equivalent-tld.          in     dname        tld.
```

This record would be added to the DNS root zone. All root name servers would have to run DNS server software capable of supporting DNAME. As of this writing, not all root servers comply. However, these upgrades can be expected within the next year, considering the natural progression of software upgrades combined with other upcoming DNS features (namely, DNSSEC).

Query load

Perhaps the most significant aspect of the proposed use of DNAME for TLD equivalence mapping is an increase in query load to the root name servers. To explain, one must understand however DNAME-capable name servers treat non-DNAME capable resolvers. Please see Appendix 4 for a pictorial description of a DNS query involving DNAME.

Consider a user wishing to visit the web sit for *www.domain.equivalent-tld.* The user's web browser asks the local resolver, or DNS client, to find IP addresses for *www.domain.equivalent-tld*. The resolver composes a DNS query for this information and sends it to the configured recursive name server (which is a combination of a name server and an iterative resolver). The recursive name server follows the standard DNS resolution process: In the absence of any cached information from previous queries, the recursive name server first sends the query for *www.domain.equivalent-tld* to a root name server. The root name servers hold a DNAME record aliasing *equivalent-tld* to *tld*, and the response returned depends on whether or not the querying recursive name server is DNAME capable.

A DNS protocol extension called EDNS[10] allows DNS protocol speakers—such as resolvers and name servers—to advertise their capabilities to each other. The DNAME specification states that DNS implementations supporting an EDNS version number

[10] Please see RFC 2671, *Extension Mechanisms for DNS (EDNS0)*

greater than zero are presumed to understand the DNAME specification in its entirety and be DNAME capable.

If the query from the recursive name server contains EDNS extensions with a serial number greater than zero, the root name server knows that the recursive name server understands DNAME records. In that case, the root name server returns just the DNAME record. The recursive name server acts on this record: it now knows that because *equivalent-tld* is really an alias for *tld*, it should "restart" the query process and now search for IP address information for *www.domain.tld*. In addition, the recursive name server caches this DNAME record and all future queries for domain names ending in *equivalent-tld* are immediately understood to be queries for *tld*: no additional DNS queries are generated to the root authoritative servers.

Unfortunately, there are few iterative resolvers that are fully DNAME capable. The iterative resolver in the BIND 9 name server does cache DNAME records and act on them appropriately. (Though, strictly speaking, there can be no fully DNAME-capable DNS implementations at the present time, since EDNS with a serial number greater than zero has not been defined.)

If the recursive name server is not DNAME capable, the resolution process is slightly different. The root name server assumes that the querying server does not understand DNAME. Rather than return an incomprehensible DNAME record, the root name server *synthesizes* a CNAME record "on the fly". This CNAME record is specific to the particular query and therefore only applicable in this instance. For example, the CNAME synthesized for the query under discussion would be this:

```
www.domain.equivalent-tld.    in    cname       www.domain.tld.
```

The recursive name server receives the CNAME and performs the same kind of "query restart" described above: it now knows that is must send a new query to resolve *www.domain.tld*. The CNAME record is cached, but note that it applies to only one domain name: it is an alias that maps only one domain name, *www.domain.equivalent-tld*, to another, *www.domain.tld*, and that's it.

Consider a subsequent query by another resolver for IP address information for *www.domain2.equivalent-tld*. A DNAME-capable recursive name server has cached the DNAME and can immediately rewrite the queried domain name to *www.domain2.tld*. But a recursive name server that doesn't understand DNAME has no choice but the query a root name server again, which will synthesize another "customized" CNAME record, this time looking like this:

```
www.domain2.equivalent-tld.    in    cname       www.domain2.tld.
```

Thus the issue is that non-DNAME-capable recursive name servers will send a query to the root name servers every time they are asked to resolve a domain name ending in *equivalent-tld*. We can expect in increase in query volume to the root name servers.

Fortunately, the query capacity of the root name servers is greatly increased in recent years as a result of widespread deployment of anycast. This technique allows multiple identical copies of an Internet resource—in this case, a root name server—to be distributed throughout the Internet to increase capacity and reliability. About half of the root operators are using anycast to increase the number of root servers. As of this writing, there are over 100 root server instances worldwide, offering tremendous excess capacity in the root server system. Obviously any proposal with potential to increase traffic to the root name servers requires careful consideration.

Memory usage

A recursive name server stores its cache of DNS resource records learned from previous resolutions in memory (as opposed to, say, a file on disk). A non-DNAME-capable recursive name server that is asked to resolve many different domain names ending in *equivalent-tld* will send many queries to the root name servers. These queries will each result in a CNAME record being returned, which the recursive name server would normally cache. Caching these records would consume memory on the recursive name server. However, a cached CNAME is unlikely, since current name server implementations that support DNAME return synthesized CNAME records with a TTL of zero seconds and therefore these records will not be cached.

It is a general principle in DNS that there is a trade off between network utilization and memory utilization. DNS administrators can adjust the "time to live" (TTL) values of records in the zones under their control. For a popular domain name (one that is queried frequently), if authoritative name servers return resource records for the domain name with a high TTL, network traffic will be reduced, since recursive name servers will store the records in their caches for a longer period of time. On the other hand, a short TTL means less time in name servers' caches and therefore fewer "cache hits" and more queries for the domain name.

If the traffic to the root name servers increased to undesirable levels, it would be possible to increase the TTL value on the synthesized CNAME records in an effort to reduce the traffic by caching these CNAME records for a short time on recursive name servers throughout the Internet. (Note that this change would require modifying the DNS server software in the case of BIND 9.) The TTL would have to be chosen as a balance between network utilization and memory consumption. A reasonable attempt could be made in advance to choose an appropriate value, but only real-world experience would allow this value to be tuned.

VI. Policy Issues and Assumptions

Two critical questions with regard to DNAME equivalence mapping for TLD strings are:
1. What policies should be established regarding DNAMEs?
2. Who should be responsible for controlling and managing DNAMEs for a given TLD?

Before delving directly into these two questions, it is important to understand the following:

1. The discussion of policy issues in this section assumes that any DNAME plan will comply with any Internet standards developed in the technical community for DNAMEs. Technical issues with regard to DNAMEs are discussed elsewhere in this paper so they will not be included here, but it should be understood that in any DNAME solution, policy and technical requirements should be synchronized.
2. DNAME equivalence mappings for TLD strings require entries into the root zone file and all changes to the root zone file require approval by the U.S. Department of Commerce (DoC). Such approval is obtained via requests sent to the IANA.

What policies should be established regarding DNAMEs?

This issue is clearly a domain name topic and as such falls into the realm of the ICANN Generic Name Supporting Organization (GNSO) for gTLDs and the Country Code Name Supporting Organization (ccNSO) for ccTLDs. The ICANN Bylaws contain a policy development process (PDP) for the GNSO and a similar process is currently being finalized for the ccNSO. The approach recommended here is to proactively deal with policy concerns from the beginning and thereby facilitate any policy development work that might need to occur in the future.

Establishing key policies before the introduction of DNAMEs would make the acceptance of their use significantly easier. To do this in a reasonable amount of time would necessitate gaining the support of major stakeholders early in the process. A recommended approach for achieving this objective would be to do the following:

- Identify key policy areas
- Identify who the major stakeholders are
- Define the key concerns of the major stakeholders with regard to all key policy areas
- Develop draft policies that will accommodate the key concerns of the major stakeholders

- Discuss the draft policies with the major stakeholders to obtain their input
- Modify the draft policies to address input from stakeholders.

There are couple different scenarios in which the above steps could happen. Once there is enough interest to pursue the DNAME solution, the GNSO and ccNSO could be contacted to find out whether they would be willing to initiate the policy work and if so how long it would be before they could start. If they could start work relatively quickly, the steps above could be performed as part of the PDP processes in each of the supporting organizations or as part of a joint PDP involving both organizations. If they were not able to commence work in a timely manner, interested parties could perform some of the steps above to lay some groundwork for the PDPs once they commence, hopefully then shortening the PDP process(es). There is also the possibility that one of the supporting organizations may not be interested in working on a DNAME solution; in such case, a solution could possibly be considered for just one category of TLDs, gTLDs or ccTLDs alone.

Key policy areas at a minimum should include:

- The selection and approval process for DNAME entries
- The role of registries and registrars (gTLD and ccTLD)
- The protection of intellectual property
- Dispute resolution
- The role of governments.

Each of these areas is discussed briefly below.

<u>The selection and approval process for DNAME entries</u>

Before a DNAME can be used for a given TLD, language/script equivalents must be assigned for TLD for each language/script community in which the registry wishes to provide registration services using DNAMEs. A couple policy questions that need to be answered in this regard are: 1) Should each registry be allowed to select its own language/script equivalents? 2) Should local communities using a common language/script have input into what language/script equivalents are used by registries and, if so, what community would be authoritative for a language/script that is used in different regions of the world? Whether or not uniform policies need to be set regarding these questions can be debated. But it would certainly seem to be sound business practice for registries to consult with local communities before selecting language/script equivalents; this would likely be a good area for market research by registries.

The approval process for DNAME entries could be as simple as submitting a request to IANA for entry of the delegations into the root zone. If global policies are established for DNAMEs, there might need to be a policy compliance check before submitting delegation requests to IANA.

The role of registries and registrars (gTLD and ccTLD)

Both gTLD and ccTLD registries would perform an essential role in implementation of DNAME TLD equivalents. They would be the primary proponents of specific DNAME TLD proposals and would therefore have the responsibility of doing any necessary market research, consulting with local communities, complying with any global policies that may be established, implementing the solution in their systems, coordinating with and educating registrars (and in some cases end users themselves if a registrar model is not used), and submitting root zone change requests to IANA.

It seems that the role of registrars and registrar resellers would not change very much from what it is today. The use of DNAME equivalents for TLDs could quite possibly expand the demand IDNs so registrars and resellers may find it useful to develop new marketing programs for IDNs. There should be very minimal demand for changes in registrar registration systems unless they are not already offering IDNs, in which case they would need to add IDNs to their product offerings.

Because of their critical relationship with domain name registrants, it will be very useful for registrars to be an active part of any DNAME policy development efforts. They also will likely be excellent sources of market demand information that registries can use in making decisions about possible deployment of DNAME TLD equivalents.

The protection of intellectual property

It is possible that DNAME TLD equivalents would have little if any impact on intellectual property protection. Because DNAME equivalents would only be used for TLDs, intellectual property issues for second level names would not change. At the same time, it would be important to involve the intellectual property and business communities early on in the consideration of DNAME TLD equivalents to give them the opportunity to evaluate possible impacts on intellectual property.

Dispute resolution

DNAME equivalents should not require any new procedures for dealing with domain name cyber squatting. The ICANN UDRP would apply equally whether or not DNAMEs are used to represent TLDs.

There is the possibility though that a new type of dispute might arise: use of a TLD DNAME by someone other than the registry delegated to provide registration services for the corresponding ASCII TLD. The issue here involves a very critical policy issue regarding the rights of registry operators for TLDs for which they have been granted registration service rights. This issue is discussed in more detail below in response to

the question, who should be responsible for controlling and managing DNAMEs for a given TLD?

The role of governments

Governments will likely have strong interest in policies related to DNAMEs. First of all, they are very interested in having fully internationalized domain names in local languages/scripts used in their countries so the possibility of implementing a solution for this that avoids some of the challenges of implementing IDN.IDNs in the DNS could be very attractive. Secondly, some governments will probably have a lot of interest in the choice of DNAME TLD equivalents used by registries. Consultation with local communities as well as the ICANN Governmental Advisory Committee will be very helpful in dealing with DNAME policy issues.

Who should be responsible for controlling and managing DNAMEs for a given TLD?

This could be one of the most controversial and political issues with regard to DNAMEs. Some countries may want to control any DNAMEs in their primary language.
It can reasonably be expected that ccTLD registries will want control of DNAMEs associated with their country codes and GTLD registries will want control of DNAMEs associated with their TLDs.

Whereas it does seem reasonable that local communities having common languages/scripts should be given the opportunity to provide input into selection of DNAME equivalents for those languages, giving control of DNAME management to countries could create some difficult problems. For a given language/script, what country should have control? When multiple countries use the same language/script, which country should have control? For ccTLDs, country control of DNAMES might work in many cases. For gTLDs that are used globally, DNAME control by one country could result in a poor user experience for those using the same language/script in other countries.

What could be the simplest approach would be to allow registries of existing TLDs to control the DNAMEs associated with their ASCII TLD names. In this approach, each gTLD and ccTLD registry could decide for itself whether it wanted to implement a DNAME solution and, if it did, it would be responsible for that implementation assuming that any approved policies were followed. If a registry elected not to implement a DNAME solution for a particular language/script, that registry could possibly give permission to another registry to offer a DNAME solution.

Some of the advantages of this approach are:

1. Existing registries would already have the infrastructure in place to perform the services necessary for implementing a DNAME solution and would merely have to modify their systems to support DNAMEs. It would not be necessary to create new TLDs and to establish registries for those TLDs in order to allow registrants to use TLD names in their own languages.
2. Disputes involving DNAMEs would be handled in the same manner and by the same organizations as those involving ASCII TLDs. It would also make it easier to deal with disputes involving similar and/or identical names registered as both ASCII and internationalized names using the DNAME functionality.
3. It would be relatively easy for registries to provide centralized information with regard to DNAMEs that correlate to ASCII TLDs.
4. All existing registries would have the opportunity to play in the DNAME game.
5. Governments could readily influence the DNAME implementation within their own ccTLD registry if desired.
6. A DNAME equivalence mapping solution could quite probably be brought to market in relatively quick order from a technical point of view.

There are at least three possible negative consequences if organizations other than a delegated registry operator are allowed to offer DNAMEs for that registry operator's TLD: 1) user confusion; 2) dispute resolution complication; and 3) TLD brand dilution. To understand these three issues, consider this example:

Domain name: IDN.tld where IDN represents a domain name in local script and .tld is in ASCII.

If the registry operator for .tld also provides a DNAME equivalent for .tld, then:

- Users of the domain name would have the same experience whether they use the ASCII or DNAME version.
- If there is a dispute, then the same registry would be involved whether it involved the ASCII or DNAME versions or both.
- The registry for .tld has full control of its brand.

If a different registry operator provides a DNAME equivalent for .tld that is either different from the one provided by the .tld registry operator or provides a DNAME equivalent for .tld in a case where the .tld registry operator does not offer a DNAME equivalent:

- Users of the domain name could have very different experiences depending on whether they used the ASCII version or one of the DNAME versions.
- If there is a dispute, then it is likely that at least two registries would be involved, thereby seriously complicating the dispute resolution process as well as implementations of any decisions made in that process.

- The .tld brand has been diluted. Registries invest considerable amounts of money and resources in developing the brand for their TLDs so it is important that that investment not be compromised.

VII. Proposal and conclusion

As described above, there exists within the DNS the capability to provide a truly internationalized service to the global internet community and to do so within existing regulatory frameworks and based upon IETF technical standards. The ICANN should study this approach and coordinate with existing TLD registries to establish a test-bed for modeling performance issues and to determine the production feasibility of this design. Specifically, the ICANN should:

1. Encourage technical feasibility and data-gathering to model system performance and impact to DNS utilizing existing TLD registry operators.

2. Establish a coordination process amongst TLDs to develop ACE assignments by language market for each TLD.

3. Establish a timeframe within which to complete any technical reviews or policy assessments.

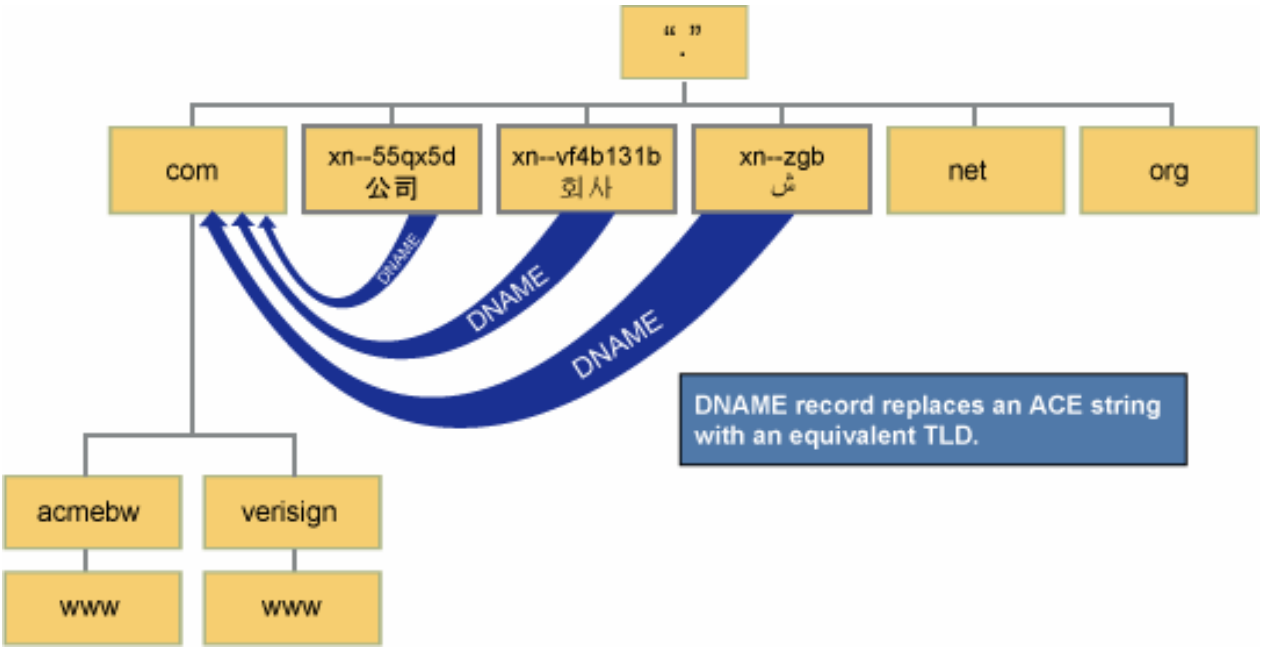
Appendix 1 - ACE examples

gTLD	Transliteration	Script/Language	Example gTLD Equivalent	punycode
com	company	Simplified Chinese	公司	xn--55qx5d
		Traditional Chinese	公司	xn--55qx5d
		Japanese	会社	xn--6oq404h
		Korean	회사	xn--vf4b131b
		Arabic	ش	xn--zgb
		Swedish	affärshus	xn--affrshus-2za
net	network	Simplified Chinese	网?	xn--ur0a138b
		Traditional Chinese	網絡	xn--od0alg
		Japanese	ネット	xn--9ckkn
		Korean	통신	xn--zv4b74y
		Arabic	ك	xn--fhb
		Swedish	nät	xn--nt-via

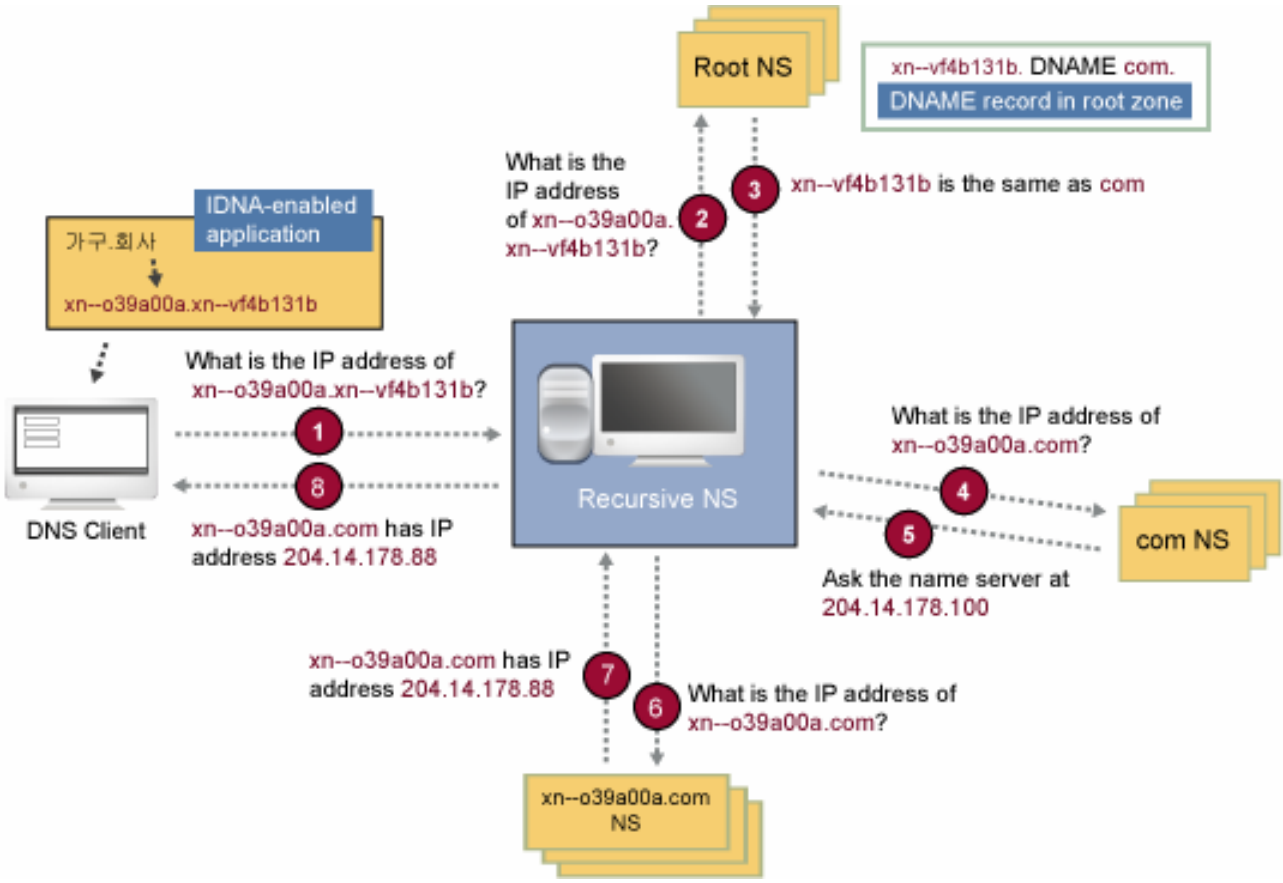
Appendix 2 – Sample DNAME entry in the root zone for “회사” (Sample Equivalent of COM in Hangul)

```
xn--vf4b131b.          DNAME      com.
```

Appendix 3 - DNS tree with DNAME equivalents



Appendix 4 - DNAME Query diagram



VeriSign®

Where it all comes together.™