INTERNET-DRAFT
Document: draft-hall-dm-idns-00.txt
Expires: May 2002

Eric A. Hall, Editor
Consultant

November 2001

The Internationalized Domain Name System

Status of this Memo

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

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

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

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

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

1. Abstract

The principle intention of this specification is to facilitate the deployment of a completely internationalized domain name syntax and service which new protocols, applications and host systems can use, but without disrupting the existing infrastructure. Towards that end, this document describes a series of elective encapsulation services and protocol extensions which cumulatively allow internationalized domain names to be stored and transmitted in the existing DNS message and within application data streams, according to the compliance level of the participating systems.

Table of Contents

2. Definitions and Terminology

This document unites, enhances and clarifies several pre-existing technologies. Readers are expected to be familiar with the following specifications:

[AMC-ACE-Z] <draft-ietf-idn-amc-ace-z>, "AMC-ACE-Z version 0.3.1"

[NAMEPREP] <draft-ietf-idn-nameprep>, "Preparation of Internationalized Host Names"

[STD13] (RFC 1034) "Domain names - concepts and facilities", (RFC 1035) "Domain names - implementation and specification"

[STD3] (RFC 1122) "Requirements for Internet Hosts -- Communication Layers", (RFC1123) "Requirements for Internet Hosts -- Application and Support"

[BCP18] (RFC 2277) "IETF Policy on Character Sets and Languages"

[RFC2279] "UTF-8, a transformation format of ISO 10646"

[RFC2671] "Extension Mechanisms for DNS (EDNS0)"

The following abbreviations are used throughout this document:

UCS (Universal Character Set) - The ISO/IEC 10646 character set repertoire, as represented by the Unicode 3.1 specification.

ACE (ASCII-Compatible Encoding) - A transfer encoding which encodes UCS character codes into a seven-bit codespace which is compatible with US-ASCII.

UTF-8 (UCS Transformation Format, Eight-Bit) - A transfer encoding which encodes UCS characters into an eight-bit codespace which is compatible with DNS message formats.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

3. Introduction

The domain name system (DNS) [STD13] currently defines a message, namespace and protocol. Although the DNS message is capable of transferring eight-bit character codes as protocol data, applications are currently limited to a subset of US-ASCII when they interact with the DNS namespace, and this restricted syntax is enforced by almost every TCP/IP application and protocol which utilizes domain names as embedded data (including, surprisingly, the DNS protocol).

In order to allow for the use of a larger range of characters in the namespace, this document extends and clarifies a variety of Internet specifications so that characters from the Universal Character Set (UCS) [ISO10646] may be used in domain names. This document also extends the DNS message structure to allow for the use of UTF-8 [RFC2279] encoded characters for the purpose of transferring these domain names, but also provides an ASCII-compatible encoding (ACE) [AMC-ACE-Z] of these character codes which existing protocols and applications can use to access the internationalized domain names, and also provides identification mechanisms which allow the end-point systems to downwardly negotiate when needed. Finally, this document defines behavior for DNS systems which implement this architecture, including the end-point applications which generate and store DNS domain names, and the resolvers, caches and servers which process them.

The mechanisms presented here are elective. Developers, zone administrators and network operators who wish to make use of the internationalized domain names may do so according to their own schedule. Those developers, administrators and operators who cannot or prefer not to implement the specified extensions can continue to use their legacy systems, and will still be able to access resources from the internationalized domain name system.

3.1. Background

From one perspective, DNS is already an "eight-bit clean" system, in that the structured DNS message is capable of storing and transmitting eight-bit data without any additional effort. However, this perspective only considers one particular facet of the domain name system, and ignores the more critical aspect of

the DNS namespace, which has rules that are entirely different from those which govern the message format.

The DNS namespace (or more appropriately, the view of the namespace which applications use and enforce) is governed by rules set forth in RFC952 [RFC952], STD3 [STD3], and STD13, which collectively define the characters that are eligible for use with host names. These rules are meant to provide a common template which may be applied to either the DNS namespace or a local hosts database, such that a query for "host.example.com" can be processed through either system. The range of valid characters currently defined are the letters, numbers and hyphen characters from US-ASCII [ASCII] (additional rules also govern the valid order and length of a host name). Character code values outside of this range are valid in domain name messages, but are undefined when used in the namespace, and are subject to interpretation by the applications which generate them.

The host name rules are enforced by almost every application and protocol which uses DNS to identify a host or system. This includes network utilities such as ping and traceroute which simply identify systems by name, and complex protocols such as SMTP which use domain names to determine message-routing paths. Portions of the DNS protocol itself are also affected by these restrictions, such as the domain names which may be used for NS resource records with sub-domain delegation operations (since these servers are connection targets, they are also required to be compliant with the host name rules).

Because these domain names are so pervasive throughout the Internet (and even within proprietary applications that run on private networks), it is not possible to declare a "flag day" at which eight-bit domain names will be considered valid encodings of a particular character set. Instead, an extended namespace with a larger set of charset rules must be defined, an extended DNS protocol capable of supporting these domain names must be deployed, and a transitional mechanism which allows the old and new systems to interact must be established. This document attempts to meet these objectives.

3.2. Objectives

In broad terms, this document has one overall goal, which is to facilitate the creation and use of an internationalized domain name system around a UCS namespace, a collection of UTF-8 and

legacy-compatible encodings which are suitable for transferring internationalized domain names within DNS and the affected application data streams, and a negotiation mechanism which allows end-point systems to identify the encoding that they will use for a particular operation.

One of the objectives stated above is to internationalize the existing DNS namespace, by allowing UCS characters to be used in host names and sub-domain delegations in old and new zones equally. As such, this document does not define a new namespace, but instead defines mechanisms by which leaf-nodes and sub-domains may be created within the existing hierarchy.

UTF-8 was chosen as the primary transfer encoding of these domain names for several reasons. For one, there is a wide availability of tools and expertise surrounding UTF-8, and it is already widely deployed within development environments, operating systems and applications. Furthermore, BCP18 [BCP18] requires that new application protocols be able to use UTF-8 as application data, and for many applications, this specifically means domain names which are passed as data. All signs indicate that UTF-8 is currently and will continue to be the preferred eight-bit encoding on the Internet, and this specification embraces this position in its design.

However, most of the network services currently in use are bound by the legacy host naming restrictions, and those applications and protocols will also need to be able to interact with resources from the internationalized namespace, even though they will not be compliant with the UTF-8 encoding mechanisms defined in this document. In order to allow these systems to participate, this specification also embraces the use of ACE as a seven-bit backwards-compatible encoding for legacy systems to use.

Note that even though a single encoding could have been specified by this document, past and present requirements would not have been satisfied by a single choice. For example, supporting UTF-8 alone would mean isolating legacy systems from resources in the UCS namespace, while supporting ACE alone would not have provided a truly internationalized namespace (the ACE encoded domain names still appear in user data quite frequently). By allowing the UTF-8 and ACE encodings to coexist, the existing and emerging communities can both be served.

Because both encodings will be active during the same time period, this document also defines DNS protocol extensions which allow the

end-point systems to detect the encoding that is in use for a particular query/response pair. Note that these negotiation mechanisms not only allow new and legacy systems to interoperate, but they also provide a transition service for developers, zone administrators and end-users, in that ACE encoded domain names can be initially deployed within existing applications and DNS systems, while individual elements of the infrastructure can be upgraded without disturbing other components.

3.3. Common Usage Scenarios

Discussion of the mechanism provided by this document depends upon the usage context of the domain names themselves. Domain names are extremely pervasive, and are used by almost every TCP/IP protocol and application in one form or another. However, most usages fall under one or more of the following scenarios:

* Connection identifiers. Domain names are most commonly used as host-specific identifiers for outbound connection requests, whether this be for a command-line application such as ping, or as a host name which is stored in an application's configuration file. Another common usage scenario for connection identifiers is with reverse lookups, where a server is logging incoming connections by the corresponding domain name, or where a program such as netstat is displaying all of the application sessions which are currently active on a host. In both of these cases, domain names are passed through applications to a resolver, resulting in DNS queries and responses which eventually provide the requested DNS data.

  A related use (but one which does not generate DNS messages) is determining the host name of the local system. This is commonly found with applications and protocols that need to display the domain name of the local system as part of a protocol operation (such as an SMTP greeting banner) or as application data.

  Connection identifiers (and lookups in general) are probably the largest single use of domain names today, and this is likely to be the case with internationalized domain names as well. This document fully supports the use of internationalized domain names for lookup operations, as long as the calling application, the stub resolver, the local caching servers, and the authoritative servers for

the specified domain name are compliant with this specification. If any of these components are not capable of supporting internationalized domain names in this manner, the ACE equivalent domain name will be negotiated for the operation at hand.

* Protocol data. Some application protocols exchange domain names as protocol data, with those domain names either determining or altering a service-specific operation. Examples of this usage include SMTP envelopes ("RCPT TO <user@domain.dom>") where the domain name is used to determine whether or not a particular email message should be accepted for delivery, the HTTP HOST header field which identifies a specific document tree on a shared server, BOOTP/DHCP options, WHOIS input, and more.

  Because these protocols treat domain names as protocol data, most of these protocols also have specific formatting requirements which must be addressed before UTF-8 domain names can be used by these protocols directly. This document is intended to facilitate the use of UTF-8 encoded domain names in this manner, although it is expected that most of the protocol development groups will need to develop negotiation mechanisms before these protocols can use internationalized domain names directly. Until such work is completed, ACE equivalent domain names can be used to provide these protocols with access to the internationalized namespace.

* Structured application data. Structured application data is similar to protocol data in that it can trigger or affect some protocol action, although this will not always occur. For example, a web browser can process an embedded IMG link which may be present in a web page, while a user can manually follow an embedded email link which is also stored in the same web page; even though both usage models share the same structured data format (URLs), they are processed differently by the application. Similarly, email messages typically contain multiple domain names as structured data in the message headers, and some of these domain names will directly affect subsequent protocol operations, while others will not.

  Because of this ambiguity, this document defines no specific treatment for structured application data. In some cases, no additional mechanisms will be required, while

other scenarios will require negotiation mechanisms before an internationalized domain name can be used in the structured data (with ACE being required as the interim format). Each protocol development group is encouraged to analyze each usage independently, to classify the usage as a connection identifier, protocol data, or unstructured application data, and to determine the appropriate course of action for each usage accordingly.

* Unstructured application data. Many application protocols provide free-text data which can contain domain names, but with those domain names existing as unstructured data. For example, an email message which is provided as a text/plain MIME body part may contain a domain name which identifies a system or service in the context of a specific application, but in an unstructured form ("your files were moved from server1 to server2"). Similarly, an email address may be provided in WHOIS output, but as unstructured data which does not affect the protocol.

  Given the application-specific nature of this data, it cannot be managed by any global protocol or process. Where a protocol has rules or restrictions on the data itself, then those rules are maintained, but some formatting rules may need to be extended before internationalized domain names (or their equivalents) can be encoded in the application data. For example, internationalized domain names in email messages may need to be converted to a preferred display charset, while ACE equivalents may be necessary for protocols which only support US-ASCII.

Each of the above scenarios represent distinct handling cases where internationalized domain names may or may not be used directly. In some cases, the internationalized domain names may be used as soon as the applications and resolvers are configured to use them, while in other cases, measured and cautious deployment is required in order to prevent undue breakage. In the latter cases, however, the backwards-compatible ACE encoding is available so that the internationalized domain names can be used.

3.4. User Audiences

Another perspective on the changes which will result from deploying the mechanisms described in this document can be seen by analyzing how any such changes will affect the different

"audiences" who work with domain names, and who have their own unique context-specific usage requirements and objectives. The three main audiences discussed in this document are:

* Developers. Protocol and application developers need to be able to incorporate internationalized domain names into their systems as easily as possible, although there are many factors which will affect such usage, including the input and output charsets and encodings which are available to the applications and protocols. Where feasible, this specification allows developers to choose any charset or encoding which may be required and suitable for use, although in most cases, a recommendation is also made for the use of UTF-8 in particular.

  Developers may adopt internationalized domain names for connection identifiers and lookup operations fairly quickly, such that users can use those system as soon as they have compliant systems (and they have a target domain name to communicate with). Implementing support for internationalized domain names in protocols and application data will require additional effort by the affected development groups.

  Support for ACE will be harder to implement, since it is a relatively new and untested encoding syntax, with no existing developer tools. This will likely be the largest hurdle to overcome when developing applications for use with this service.

* Zone administrators. Organizations that wish to deploy internationalized domain names should be able to do so easily, at a reasonable cost, and without suffering excessive pre-conditions. Towards this objective, the mechanisms described by this document allow organizations to deploy and use internationalized domain names within any zone immediately, without requiring any other zone to have been updated beforehand (although there are specific and strong suggestions for upgrading the Internet's high-load servers as soon as possible).

  If an organization wishes to publish internationalized domain names for users to access and utilize, the authoritative servers for the affected zone must be compliant with the naming rules and message formats described by this document, which will almost certainly

require the administrators of that zone to upgrade their servers. However, organizations may also choose to only deploy ACE encoded domain names if an immediate migration is not feasible, with the caveat that internationalized domain names in their native form will not be available from those zones.

* Network operators. The systems and human users which generate DNS lookups are another area of concern, as these protocols, programs and users will expect these lookups to succeed, and will also expect that the visible namespace will be compatible with the capabilities of the requesting system at a minimum investment. This is a broad range of requirements.

  At a minimum, applications must be capable of generating and accepting the internationalized domain names if they are to use those domain names (see the "Developers" discussion above for the application requirements). Similarly, the local resolvers, caches and forwarders on the user's network must also support the message formats if they are to relay internationalized domain names between their local applications and the remote zones being queried. If the applications, resolvers and caches do not support these requirements, intermediary systems will perform the down-level negotiation automatically on their behalf such that additional effort is not required on the user's part.

In summary, the developers, zone administrators and end-users can immediately participate in the internationalized namespace at no additional expense if they are content with using ACE encoded domain names, and can use internationalized domain names in their native form if they are willing to make the necessary investments. Furthermore, since the native and backwards-compatible encodings are not mutually exclusive, implementers of this specification have the option of adopting ACE for immediate use and then transitioning to internationalized domain names on a per-system, per-zone, or per-application basis, according to their schedule.

3.5. Service Overview

This document specifies a variety of extensions to several different protocols and services in order to facilitate the use of internationalized domain names anywhere this support exists or can

be implemented, and to provide a legacy-compatible domain name in all other situations.

More specifically, this document defines or clarifies behavior for the following elements:

* Host name character restrictions. Legacy protocols and applications are currently restricted to the legacy host naming rules, which only allow for a subset of US-ASCII characters (letters, digits and the hyphen character). This document redefines the characters which are valid within a host name so that system identifiers, domain name parts of host names, and new network services can use most of the characters from the UCS.

* DNS message format. This document defines an extended label format based on the extended label services provided by RFC2671 (Extension Mechanisms for DNS - EDNS0) [RFC2671], with this label format being used to encapsulate UTF-8 encoded internationalized domain names in DNS messages. Any DNS message which carries the UTF-8 encoded domain names is required to use the EDNS/UTF-8 label type defined in this document. Any DNS message which carries legacy domain names (including the ACE encoded equivalent domain names) is required to use the traditional message format.

* Application handling rules. Applications can use internationalized domain names immediately for lookup operations that do not directly affect external services or protocols, and can use ACE encoding sequences to specify internationalized domain names in legacy protocol operations, and can use them both at the same time.

* Stub resolvers. Stub resolvers will most likely need to provide a series of internationalized APIs in order to fully support applications that generate internationalized domain name lookups. For example, these APIs will almost certainly be required in order for the resolver to determine that the calling application is compliant with the host name requirements defined by this document, and that the domain names should be encoded in the proper label format. Although this specification does not dictate these APIs, it encourages their use, and provides some guidance on the issues surrounding their use.

* Forwarders, resolving servers and caches. The user-side servers which process internationalized domain names have several protocol-specific requirements, including the negotiated fall-back service when UTF-8 queries fail.

* Authoritative servers. A key part of this specification is the simultaneous support for internationalized and legacy compatible domain names in the UCS namespace, thereby allowing a domain name to be entered into an authoritative zone database once, and for the appropriate response to be generated by a server according to the label encoding from the associated query. In order for this to work, this specification requires authoritative servers which serve internationalized domain names to comply with specific conditions. This specification also allows existing servers to serve ACE equivalent domain names when the authoritative servers cannot be upgraded, although this typically results in lower levels of functionality.

The elements listed above collectively define a completely internationalized domain name system, which is capable of servicing internationalized domain names in all compliant systems, and which is also capable of providing ACE encoded equivalent domain names when any component from the internationalized service is not available.

3.6. Process Example

This section illustrates a series of query/response transactions under which the processes and protocols defined in this document function. This example uses a reverse lookup for the PTR resource record associated with the "14.2.0.192.in-addr.arpa." domain name (forward lookups work similarly, but the issues are more fully demonstrated by PTR lookups). Each of the various technologies shown below are described in later sections of this document. The sole purpose of this example is to provide an illustration of these mechanisms in order to facilitate better discussion.

Note that this illustration represents a worst-case scenario (thereby exercising most of the functionality provided by this specification), and does not represent a typical scenario.

a. First, a PTR resource record for 14.2.0.192.in-addr.arpa. is added to the internationalized zone database on the replication master server for the 2.0.192.in-addr.arpa.

zone, with the resource record data value of "host.<idn>.example.com." (where <idn> is an internationalized domain name compliant with the host naming rules provided in this document). Both of these domain names have a primary representation consisting of UCS characters in some local encoding, but are also available as UTF-8 and ACE encoded data so they can be encapsulated within DNS queries and responses.

Once the zone is reloaded and is replicated by the other authoritative servers for that zone, the domain names can be processed.

b. An application on a remote system generates a DNS lookup for the PTR resource record associated with the 14.2.0.192.in-addr.arpa. domain name.

If this is a legacy application, it issues the lookup using the only method it knows, which is to pass the domain name to the legacy resolver API. This would result in the resolver issuing a legacy DNS query for the PTR resource record associated with the specified domain name.

If this application is compliant with this specification, it performs the following steps:

1. Verify that the resolver is capable of processing queries for UTF-8 domain names by probing for an internationalized API. If this step failed, then the domain name would be converted to the legacy STD13 octet encoding in step 3.6.b.3 and passed to the resolver's legacy API.

2. Convert the domain name from its generated encoding to the canonical UCS characters, and then normalize and case-convert the UCS characters.

3. Convert the normalized and lowercased UCS characters to the charset or encoding used by the resolver's internationalized API.

4. Issue a lookup for the PTR resource record associated with the internationalized domain name, via the resolver's internationalized API.

Note that even though the domain name is compatible with the legacy host name rules, the domain name is passed through the internationalized API so that servers can tell whether or not the original application is UTF-8 compliant, and can determine the format of any internationalized domain names which are to be returned in the response messages. This is required in case the queried resource record includes internationalized domain names as resource record data (as would be the case with PTR resource records), and is also required for the proper handling of any SOA or NS resource records which may be returned as additional data in the response.

For the purpose of this example, we will assume that each of these steps were successfully performed.

c. The client's stub resolver generates the query, with the Question Section of the query containing the UTF-8 encoded domain name encapsulated in an EDNS/UTF-8 extended label.

d. The stub resolver sends the query to one of its configured resolving servers.

e. The resolving server will either answer the query from its cache or forward the query to a name server which is authoritative for the namespace hierarchy, as per the normal query-resolution procedure. For the purpose of this example, we will assume that the server has no information about the specified domain name, so it forwards the query to one of the root zone's authoritative servers in order to begin the iterative resolution process.

f. The queried server responds with a referral, providing delegation data for a zone in the path to the queried domain name. For the purposes of this example, we will use 192.in-addr.arpa. as the delegation domain specified in the referral message.

The specific format of the referral will depend on whether or not the queried server understands the EDNS/UTF-8 label encoding. If the server is compliant with this specification (which it is, or else it wouldn't have answered with a referral), then the referral will also provide ENDS/UTF-8 encoded domain names in the Authority and Additional-Data Sections of the referral. If the server

was not compliant with this specification, it would return an error upon seeing the extended label type, which would cause the resolving server to restart the query using the legacy label type.

g. The resolving server decodes the UTF-8 encoded domain names to their UCS character representation, caches the resource records in their UCS form, and sends the query to one of the authoritative servers for the referral zone. Note that the cache did not normalize or case-convert the UCS characters; only the end-systems perform this work.

h. In this case, the queried server does not understand the EDNS/UTF-8 label format, and has returned a FORMERR response code.

i. When these errors are encountered, the current resolver (whether this is the client's stub resolver or a caching server in the query path) must convert the query domain name from its current form to a legacy-compatible encoding (either ACE or STD13 octet sequences, depending on the UCS characters which have been encoded), and then has to reissue the query in that format.

   In this case, the domain name only contains printable characters from US-ASCII, so the STD13 octet encoding is used for the fall-back query. Because the UCS domain name was normalized and lowercased before it was passed to the client's stub resolver, the legacy domain name will also be in this format (although it will be compared in a case-neutral form by the recipient server).

   Note that once this conversion takes place, the legacy label format is used for the remainder of the current query chain (this prevents excessive delays from multiple fall-back operations, which could result in timeouts at the original resolver or application).

j. The queried server returns a delegation referral for the 2.0.192.in-addr.arpa. zone. Since the query arrived in the STD13 octet encoding, the server has no indicator of the client's capabilities, so the referral NS resource records will also be returned in legacy compatible form (either as STD13 octet sequences or as ACE encoded data, depending on the character codes provided in each label from each of the associated domain names).

   Note that even though these NS resource records will be restricted to legacy-compatible host names and label types, they may contain and reference ACE domain names. In this regard, a legacy server in the delegation path does not prevent internationalized domain names from being delegated or resolved, but only prevents them from being processed as EDNS/UTF-8 extended labels.

   Also note that once the authoritative servers for a zone have been discovered and cached, any subsequent UTF-8 queries which are generated for the resources in that zone will be sent directly to one of those servers, bypassing the delegation hierarchy. As such, subsequent queries which are provided in EDNS/UTF-8 labels can be processed directly by the zone's authoritative servers, without the delegation servers disrupting the process.

k. The resolving server decodes the STD13 octet sequences and ACE encoded domain names to their UCS character representations, caches the resource records, and resends the query to one of the authoritative servers for the referral zone.

l. The queried server processes the request. Since this query arrived as an STD13 octet sequence, the server must compare the seven-bit characters from the domain name (which is all of them, in this example) in a case-neutral form. Note that if the query had arrived as ACE or UTF-8 encoded domain names, the server would have decoded the specified domain name to its canonical UCS characters and performed a case-exact match against the resulting characters.

m. The queried server responds with the requested data. Note that the query was submitted in the legacy label form due to the fall-back processing which occurred in step 3.6.i, so the server will only respond to this query with STD13

octet sequences or ACE encoded domain names, using the STD13 legacy label.

n. The resolving server decodes the STD13 octet sequences and ACE encoded domain names to their UCS character representations, and caches the resource records. Since the query was originally received as an internationalized domain name (as indicated by the EDNS/UTF-8 extended label from the original query), the resolving server has to encode the answer data as UTF-8 before passing it back to the client's stub resolver. However, since the input was not provided in an encoded UCS form, the server has to normalize and case-convert the STD13 octet sequence in order to provide a valid internationalized domain name.

o. The stub resolver decodes the UTF-8 encoded domain names which have been provided in the response message to their UCS character representation, and passes the data to the original calling application using the charset or encoding favored by the resolver.

p. The application validates the received domain name by decoding the internationalized domain name to its canonical UCS characters, normalizing and down-casing the resulting domain name, and comparing the results with the answer data which was provided by the resolver.

As can be seen, the UTF-8 name resolution process is identical to the current resolution process, with the addition of a single fall-back query in step 3.6.i which resulted in one extra query/response pair (roughly equivalent to adding one extra delegation referral into the query path), and with several different encoding conversions, as required by the participating systems and services. This example also illustrates the requirements which are placed on developers, zone administrators, and network operators in order for typical connection identifier services to function with UTF-8 domain names.

However, if each system and service had used UTF-8 for encoding purposes (including everything between the stub resolver's APIs and the authoritative servers for the target zone), then no additional queries or conversions would have been required (other than the direct UCS conversions required for validation and caching, the latter of which can be performed separately without affecting the processing path). In this regard, the example above illustrates how this system can function even when only a portion

of the participating systems utilize UTF-8, and also illustrates how effective the entire operation would be if all of the recommendations and requirements provided in this specification were adopted.

It is also important to reiterate here that any such costs associated with this compliance are entirely elective by the affected parties. If they want to streamline the process, the option is available to them, although the system also works when very few optimizations are implemented.

4. The Internationalized Namespace

In simple terms, this specification defines an internationalized namespace which consists of domain names and labels that contain UCS character codes, and also specifies a series of encoding formats which may be used whenever the UCS values need to be encapsulated for transmission within DNS messages or application data streams.

In this regard, the internationalized namespace is the UCS representation of the domain names and labels as they are used for comparison operations once a domain name arrives for processing, while the transfer encodings ensure that a domain name arrives at the destination system intact, so that it may be processed in its canonical form.

There are four conceptual elements to this model:

* Character codes. Labels from internationalized domain names have a single logical canonical representation as sequences of UCS code point values. The UCS characters are used when a particular label from a domain name is created by an application, stored in a zone, hosts or cache database, and is used whenever two sets of domain names or labels need to be compared. However, different kinds of domain names have different rules which govern the character codes that may be used.

* Storage encodings. Whenever a domain name is created or copied from the network, it must be stored in a format that is reversible to the canonical UCS character representation of that domain name. This specification does not mandate or require any particular storage encoding, and allows this decision to be made on a per-implementation basis, as long

as the storage encoding supports character codes which can be converted to UCS equivalent values for comparison purposes. However, the use of UTF-8 for this purpose is encouraged, since it is the most common.

* Transfer encodings. Whenever a domain name needs to be sent over the network, it must be packaged in a form which is compliant with the capabilities of the transfer protocol in use. This document specifies three transfer encodings which may be used to encode canonical UCS character codes in DNS messages or application streams, which are: the octet encoding from STD13, the ACE encoding from <ACE-Z>, and the UTF-8 encoding from RFC2279. Each encoding has different costs and benefits in different usage scenarios.

* Comparison operations. When two domain names need to be compared, they also follow rules which are appropriate to the type of domain name being provided, and the transfer encoding which may have been used to provide the domain name to the system.

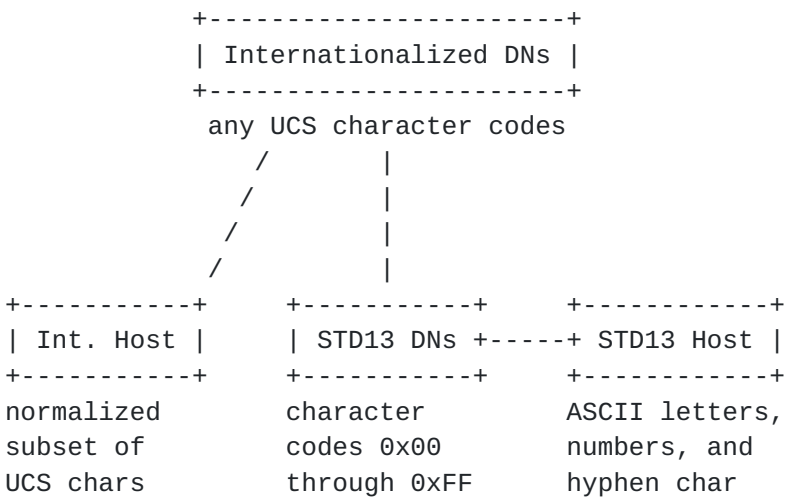
This document defines four distinct types of internationalized domain names which may exist in the internationalized namespace, and also describes how each of the above considerations affect those domain names and their labels. These domain name types are described throughout the remainder of this section.

4.1. Internationalized Domain Names and Labels

This section describes the master template rules for all domain names and labels which may be used in the internationalized namespace, although subordinate rules and restrictions are also applied as secondary filters, depending on the intended usage of the domain name.

For example, domain names and labels which are to be used as internationalized host identifiers (either as host names, or as domain names which are used to specify a host) are restricted to a specific subset of UCS characters. Meanwhile, domain names and labels which are compliant with STD13's global rules are restricted to eight-bit code values, while the domain names and labels which are used as STD13 host identifiers are restricted to a specific subset of US-ASCII.

The following diagram illustrates how the subordinate rules are applied and interpreted against the master restrictions:



As can be seen, the internationalized domain names and labels rules allow any UCS character code to be stored, although each particular usage of the domain names and labels will have their own secondary rules and restrictions.

In order to allow future documents to define additional rules as required for their usage, this document defines very few global rules on the core internationalized domain names and labels.

4.1.1. IDN syntax and structure

In this specification, an internationalized domain name consists of a variable number of labels, each of which contain a variable number of UCS character codes, not all of which will have defined UCS character interpretations.

Furthermore, the encoding system which is used to store and interpret those values on a system is not relevant to this specification, and is therefore not defined. The characters in a label can be stored in memory or on disk as UTF-8, UCS-4, ACE, or any other storage encoding which is desired by the operators and implementers of the affected system, as long as that encoding system is reversible to the canonical UCS character code values, and is able to represent the necessary range of UCS characters (the "necessary range" varies by operation).

The only universal restrictions which apply to internationalized domain names and labels are those which govern length. This specification requires that labels from internationalized domain names MUST be restricted to a minimum length of two characters and a maximum length of 63 characters, inclusive. The exception to this rule is the root domain, which is always represented by a zero-length label. Note that this rule specifically refers to the canonical UCS characters, rather than any encoded form (encoding will often result in labels and domain names with fewer actual characters, due to overhead from the encoding algorithm).

A fully-qualified internationalized domain name is formed by joining a series of labels together, with the most-contextually specific label in the left-most position of the label sequence, and with the root domain occupying the right-most position. The sum total of all labels in an internationalized domain name MUST NOT exceed 255 characters, inclusive. Any number of labels MAY be stored in the domain name, but the sum total of their lengths MUST NOT exceed this limit.

However, labels which contain UCS character codes greater than U+007F will result in multi-byte UTF-8 and ACE encodings, so the maximum length of a label or an internationalized domain name is governed by their UTF-8 and ACE encoded lengths. Both encodings MUST result in an encoded length of 63 octets or less in order to be usable, with a maximum cumulative length of 255 octets.

4.1.2. IDN transfer encodings

The UCS is currently occupies a 21-bit range of character code values, containing tens of thousands of assigned characters, and hundreds of thousands of unassigned characters. Due to the multi-byte nature of the code point values, UCS characters cannot be passed as protocol or application data in most of the existing Internet protocols (including DNS messages), at least not without the help of some kind of encoding scheme. At the very least, the UCS character values have to be encoded as eight-bit sequences if they are to fit within existing eight-bit data structures, and have to be encoded as a subset of US-ASCII characters if they are to be usable with legacy protocols and applications which only use STD13's host identifier rules for their structured domain name data types.

With this objective in mind, this document defines three different transfer encoding systems which can be used to convert

internationalized domain names and labels into a form which is suitable for transfer in different data streams. These are the legacy STD13 octet encoding, ACE, and UTF-8. Each of these encoding schemes provide different benefits and capabilities to the internationalized DNS effort.

* STD13 octets. The STD13 octet encoding scheme provides a direct one-to-one mapping between eight-bit characters and their eight-bit values, but it is only capable of storing character codes in the range of U+0000 through U+00FF, which severely restricts its usefulness.

* ACE. The ACE encoding scheme is capable of storing UCS character code value as seven-bit sequences in STD13 legacy labels. While this makes it practically compatible with the legacy host identifier rules, the resulting data imposes additional labor on the Internet community, and the reuse of the legacy label also results in certain amounts of ambiguity with some DNS domain names and labels.

* UTF-8. The UTF-8 encoding scheme is capable of encoding all UCS character code values as sequences of eight-bit data which are compatible with legacy DNS message restrictions, but the encoded output requires explicit support from internationalized applications and protocols. UTF-8 output uses a new label type in order to prevent additional ambiguity problems from arising.

The table below illustrates the UCS character code sequences which are supported by each of the different encoding schemes.

	STD13 Octets	ACE	UTF-8
US-ASCII	Y		Y
Eight-Bit	Y	Y	Y
Any UCS Chars		Y	Y

More specifically, the character code sequence ranges and their valid encodings are:

* US-ASCII. If a label only contains character codes from the range of U+0000 through U+007F, then it MAY be encoded as a legacy STD13 octet sequence or UTF-8, but MUST NOT be encoded as ACE.

  Note that this specification explicitly prohibits seven-bit labels from being encoded as ACE data, since such an action would be redundant, results in greater processing overhead for those labels, and multiple representations introduce problems with caches on legacy systems. Furthermore, certain security risks would be introduced if this were allowed. For example, a malicious user could register or purposefully create an ACE encoded representation of the "example.com" label sequence such that users mistakenly sent sensitive data to malicious systems.

  In order to prevent these problems from occurring, this specification requires that any ACE-encoded label which consists entirely of seven-bit characters MUST be immediately discarded with extreme prejudice. This rule applies to every implementation of this specification, including any applications, resolvers, caches or servers which process labels.

* Eight-bit codes. If a label contains character codes from the eight-bit range of U+0000 through U+00FF, then it MAY be encoded as STD13 octet sequences, ACE, or UTF-8. This rule specifically requires that the label MUST contain at least one character from the eight-bit range, MAY contain any number of characters from the seven-bit range, but MUST NOT contain characters with code values which are greater than U+00FF.

  Since the STD13 octet encoding and ACE both use the legacy STD13 label type, this specification relies on the input encoding of a domain name in order to determine the output encoding. In some cases, however, the input encoding will not be clear, or will not be specified, and this can result in some ambiguity with label sequences from this range.

  For example, if the domain name provided in a query consists of seven-bit labels, then the STD13 octet sequence is the only valid encoding for the legacy STD13 label,

meaning that ACE could not have been used in the query. If the specified domain name exists as a CNAME resource record which refers to a domain name that contains eight-bit character codes, then the proper output encoding for that domain name will not be clearly discernable. Moreover, the STD13 and ACE encodings will generate different results, since the STD13 octet sequence will only contain a single octet for the eight-bit character, while the ACE encoding will contain multiple octets of encoded data.

When this situation arises, systems MUST give preference to the ACE encoding, on the assumption that the referenced character is more likely to represent a UCS character than an eight-bit code value (the UCS characters in this range are Latin-1, which are the most common characters after the legacy US-ASCII set). Furthermore, the ACE encoded representation of these characters allow for a broader range of subsequent operations (since it complies with the legacy host naming restrictions, it can be used with CNAME resource records that refer to hosts), while the STD13 octet encoded representation does not.

It is possible to avoid this scenario on authoritative zone servers (and thus the affected caches) by allowing the operator to specify whether or not the input is Latin-1 UCS character data or binary data, with the server generating the proper output accordingly. Also note that the default encoding specified by this document is UTF-8, which does not suffer from the ambiguity problems described above.

* Any UCS character codes. If a label consists of any character codes greater than U+00FF, then it MAY be encoded as ACE or UTF-8, but MUST NOT be encoded as STD13 octet sequences. STD13 is not capable of representing character codes greater than U+00FF, so it cannot be used with any UCS characters beyond the eight-bit range.

Encodings are performed on a per-label basis. Each label MUST NOT be encoded more than once. Also note that recursive encodings result in applications discarding the domain name.

When the STD13 octet encoding is used to encode labels for transmission, the labels are encoded according to the rules specified in STD13, and are encapsulated in STD13 legacy labels.

When ACE is used to encode labels for transmission, the labels are encoded according to the rules specified in <ACE-Z>, and are encapsulated in STD13 legacy labels (this process is described in section 5.2).

When UTF-8 is used to encode labels for transmission, the labels are encoded according to the rules specified in RFC2279, and are encapsulated in EDNS/UTF-8 extended labels (the format of this label is described in section 5.1).

Note that a domain name MAY contain any combination of STD13 octet encoded labels and ACE encoded labels. However, if a domain name contains any UTF-8 encoded labels, then ALL of the labels from that domain name MUST be encoded as UTF-8 data. This rule primarily exists so that DNS compression services can be maintained consistently, but it also prevents mixed referrals which can trigger unnecessary fall-back processing, and also provides a single encoding representation to internationalized systems which benefits efficiency.

The root domain (as specified by the zero-length label at the right edge of the domain name) MUST NOT be encoded with ACE. More specifically, zero-length labels MUST NOT contain any character data of any kind, and since ACE labels have prefix strings, they are explicitly forbidden from being used for the root domain.

4.1.3. IDN comparison operations

When an internationalized domain name label is received from the network as ACE or UTF-8 encoded data, the labels MUST be decoded to their canonical UCS character representation, and the resulting UCS characters MUST be compared as case-exact sequences to their stored equivalents. Except where specifically required in this specification (EG, validity tests which are performed by applications), normalization and case-conversion MUST NOT be performed against the resulting UCS character codes prior to any comparison operations being performed.

However, internationalized domain name labels which are received as STD13 octet sequences MUST be given special treatment, as these domain names could have originated from legacy systems operating under STD13's rules. In this case, the seven-bit US-ASCII alphabetic characters (U+0041 through U+005A, and U+0061 through U+007A) from those labels MUST be compared in a case-neutral form. All other code values MUST be compared as case-exact code values

(this particularly includes eight-bit characters, which were not defined by STD13).

4.2. Internationalized Host Identifiers

Internationalized host identifiers are a subset of the internationalized domain names described in section 4.1, which only use a subset of the allowable UCS characters, but which reuse the global transfer encodings and comparison routines.

Most of the displayable characters from the UCS can be used in host identifiers, and there are no additional rules governing the ordering or length of their labels. However, the characters which are used in internationalized host identifiers MUST be normalized and case-converted before they are encoded for storage or transfer. This requires more effort on the part of applications and servers when the internationalized domain names are initially created, but results in less ambiguity and lower processing requirements for servers, caches and resolvers during subsequent comparison operations.

The restrictions which govern the creation of internationalized host identifiers are as follows:

a. Labels MUST be restricted to the subset of characters which are permitted by <nameprep> [nameprep]. Characters which are prohibited by <nameprep> MUST NOT appear in any label of any internationalized host identifier.

b. Labels MUST be normalized through <nameprep> before they are stored or encoded for transfer. Internationalized host identifiers will not be normalized as part of any comparison operation, so systems MUST normalize the labels before they are stored or transmitted.

c. Labels MUST be converted to lowercase according to the case-mappings rules specified in <nameprep> before they are stored or encoded for transfer. Internationalized host identifiers will not be converted to lowercase as part of any comparison operation, so systems MUST normalize the labels before they are stored or transmitted.

According to the rules above, a label from an internationalized host identifier which was originally created with the UCS character sequence of <LATIN CAPITAL LETTER A><COMBINING ACUTE

ACCENT><LATIN CAPITAL LETTER B> (U+0041 U+0301 U+0042) would be normalized and lowercased to <LATIN SMALL LETTER A WITH ACUTE><LATIN SMALL LETTER B> (U+00E1 U+0062). The normalized, lowercase form would be used as the canonical UCS character representation of that label when it was encoded for storage and transmission purposes, and would be the form which was used for comparison operations on any resolvers, caches and servers.

Internationalized host identifiers which are received from the network can contain labels which have been encoded as STD13 octet sequences, ACE or UTF-8. In all of these cases, the comparison rules defined in section 4.1.3 MUST be applied.

4.3. STD13 Domain Names

STD13 allows any eight-bit code values to be used in domain name labels. However, STD13 host identifiers (as described in section 4.4 of this specification) are the most common form of STD13 domain names, and have much tighter restrictions.

There are common uses of STD13 domain names which do not comply with the STD13 host identifier subset, however. One common example of this is SRV identifiers, which use an underscore character (U+005F) as part of their label syntax. Another common example is found when email addresses are provided in SOA and RP resource records, and where the left-hand side of the email address is stored as an STD13 domain name label which does not represent a host identifier. Furthermore, email addresses often contain extra characters which are not legal in STD13 host identifiers, such as a full-stop character (U+002E). For example, "joe.admin" could be stored as an STD13 domain name label in the fully-qualified domain name of "joe.admin.example.com.", which would represent the email address of "joe.admin@example.com" when that domain name was extracted from the SOA or RP resource record and processed.

Implementations of this specification MUST allow STD13 domain names to be created and stored, using the following rules:

a. Labels MUST be restricted to the code values of U+0000 through U+00FF. Restrictions on character content MUST NOT be applied (note that if this domain name will be used as part of an STD13 host identifier, the rules specified in section 4.4 MUST be used instead).

b. Labels MUST NOT be normalized or lowercased before they are stored or encoded for transfer.

c. Systems MUST allow STD13 domain names to be specified as exact sequences of eight-bit octet values, and MUST NOT treat these sequences as canonical UCS characters which are normalized or lowercased. STD13 defines an escaping mechanism whereby the decimal value of the octet is prefaced with a reverse-solidus (such as "\193"), which is suggested for this usage.

STD13 domain names which are received from the network can contain labels which have been encoded as STD13 octet sequences, ACE or UTF-8. In all of these cases, the comparison rules defined in section 4.1.3 MUST be applied. Note that some of these sequences can contain octet code values which have not been normalized or lowercased by the originating system, since these values can be used to specify binary domain names.

4.4. STD13 Host Identifiers

This document does not deprecate, replace or modify the host name rules defined by RFC952, STD3 or STD13 as they apply to legacy host identifiers. However, there are several issues which affect the usage of these domain names and their labels in this system.

The range of characters which are currently defined as valid in STD13 host identifiers are the uppercase and lowercase letters, numbers and hyphen character from US-ASCII. No other characters are allowed to be used. Furthermore, the current rules also prohibit the use of the hyphen character in the first or last character position of a host identifier label.

Implementations of this specification MUST allow STD13 host identifiers to be created and stored, using the following rules:

a. Labels MUST be restricted to the code values of U+002D, U+0031 through U+0039, U+0041 through U+005A, and U+0061 through U+007A.

b. Labels MUST NOT contain the code value of U+002D in either the first or last character position of the label.

c. The alphabetic characters MUST be converted to lowercase before they are stored or transmitted. STD13 host identifiers are always compared in a case-neutral form.

STD13 host identifiers which are received from the network can contain labels which have been encoded as STD13 octet sequences UTF-8. In both cases, the comparison rules defined in section 4.1.3 MUST be applied.

5. Transfer Encodings and Label Types

As was discussed in section 4.1.2, internationalized domain names and labels are required to be encoded as either eight-bit or seven-bit data whenever they are transmitted as protocol or application data.

The particular output encoding format which will be used for any given label will be primarily determined by the capabilities of the participating end-point systems. If the application or protocol which is relaying the domain name labels supports internationalized domain names directly then UTF-8 encoded labels can be used, but if the protocol or application is only capable of supporting STD13 host identifiers as domain name data, then the STD13 octet and/or ACE encoded labels will have to be used.

With DNS messages in particular, the "data type" is the label encapsulation in use. Although STD13 legacy labels allow for the use of eight-bit codes, multiple encodings for the same basic character data result in interpretation problems without some form of ancillary tagging service. For this reason, each encoding is represented differently by this specification. When the STD13 legacy label contains STD13 octet sequences then no tagging is provided, but if the STD13 legacy label contains ACE encoded data then the encoded sequence is tagged with an ACE identifier (a character prefix which does not normally appear in labels). When UTF-8 domain names are provided, an EDNS/UTF-8 extended label is used to encapsulate the internationalized domain name.

Furthermore, the encoding which is used for any label in the message will also determine the label type which is used to encapsulate and transfer the entire domain name. If any label contains EDNS/UTF-8 extended labels, then all of the labels from that domain name are required to be encapsulated for transfer in EDNS/UTF-8 extended labels. Conversely, if a domain name contains ACE or STD13 octet encoded labels, then all of the labels from

that domain name are required to be encapsulated for transfer using the STD13 legacy label format.

Note that other legacy applications and protocols will most likely be required to provide extended encodings or negotiation features before they can exchange internationalized domain names directly. However, new applications and protocols which are subsequently written to comply with BCP18 and this specification should not require any such effort, as they should be capable of transferring UTF-8 domain names from the beginning.

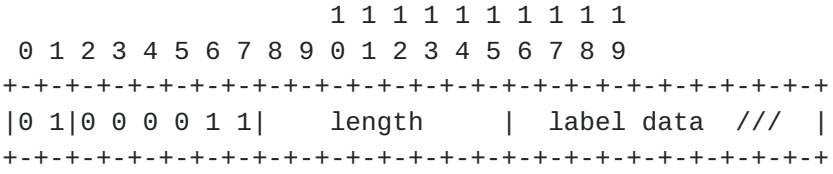
5.1. The EDNS/UTF-8 Label Type

Any internationalized domain name label which has been encoded as UTF-8 for transmission in a DNS message MUST be encapsulated as a EDNS/UTF-8 label.

The EDNS/UTF-8 extended label is an instance of EDNS extended label types (as defined by RFC2671). Extended labels are indicated by the leading bit pattern of 0b01 in the label type field (the first two bits from the "label length" octet of the STD13 legacy label type), with the remaining six bits of this octet indicating the extended label type in use. The EDNS/UTF-8 label type uses the binary value of 0b000011 for this indication (note that IANA may change this assignment).

EDNS/UTF-8 labels contain two subordinate units of data. The first octet contains a length indicator which works exactly the same as the length octet as used by STD13 legacy labels: if the first two bits of this octet are 0b00 then the rest of that octet provides the length of the label data field, but if the first two bits of this octet are 0b11 then the label is a pointer to some other label, and the remainder of the length octet provides an off-set which points to the length octet of the referenced label, as per the rules provided in section 4.1.4 of RFC 1035 (STD13, part 2).

The structure of the EDNS/UTF-8 extended label is illustrated by the following figure.



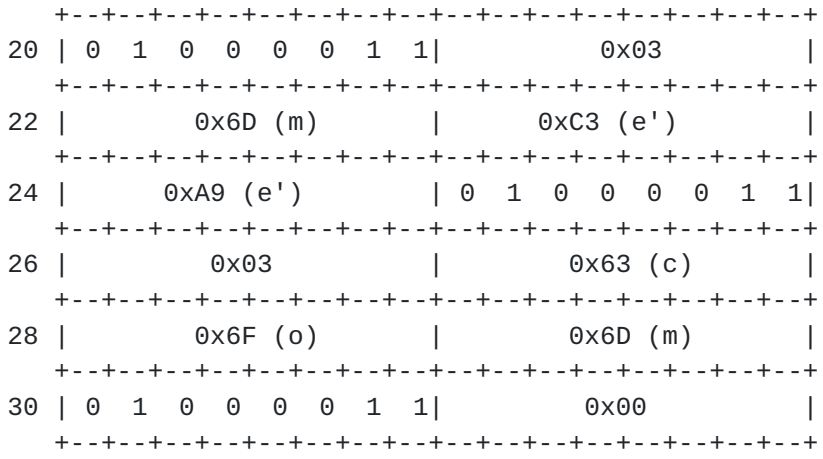
0b01 - The extended label identifier.

0b000011 - The EDNS/UTF-8 extended label type identifier.

Length - The number of octets in the label data, or the offset to the length octet of another EDNS/UTF-8 label.

Label data - The label data, encoded as UTF-8 octets.

The following example shows the domain name of me.com, where the "e" in "me" is the UCS character <LATIN SMALL LETTER E WITH ACUTE> (U+00E9), which has the UTF-8 encoded octet sequence of 0xC3A9.



Octet 20 identifies the EDNS/UTF-8 extended label type, while octet 21 indicates that the label is three octets long. Octet 22 contains the UTF-8 value for lowercase "m", while octets 23 and 24 contain the UTF-8 value for the UCS character <LATIN SMALL LETTER E WITH ACUTE> (encoded as 0xC3A9).

Similarly, octet 25 identifies another EDNS/UTF-8 extended label type, while octet 26 indicates that the label is three octets long, while octets 27 through 29 contain the UTF-8 values for the lowercase alphabetic sequence of "com".

Finally, octet 30 identifies another EDNS/UTF-8 extended label type, while octet 31 indicates that the label is zero octets in length, thereby signifying the root zone (the end of the queried domain name).

Note that the use of the EDNS/UTF-8 extended label type serves multiple purposes. On the one hand, it provides a method of signaling the resolver's capabilities to the server, so that the server can determine which format it needs to use when returning answers, referrals or errors. Moreover, using an encapsulation format which is not backwards compatible prevents certain ambiguity problems which can result from overloading the STD13 legacy label with multiple encodings. These problems are seen in certain situations with STD13 octet encoding and ACE, where a server cannot adequately determine which encoding a resolver desires. By using a separate extended label type for UT-8, these kinds of ambiguities are avoided.

There are additional benefits which come from using EDNS extended label types, which are best expressed as "future possibilities". Once the EDNS extended label mechanisms are widely deployed, it becomes feasible to specify additional encoding mechanisms as soon as the Internet community deems it desirable. In this regard, defining alternative encodings is much easier the second time.

5.2. The STD13 Legacy Label Type

Any internationalized domain name label which has been encoded as ACE or STD13 octet sequences for transmission in a DNS message MUST be encapsulated within an STD13 legacy label.

This document does not deprecate, replace or extend the STD13 octet encoding or label encapsulation rules defined by STD13. However, this document does provide some guidance on the creation and interpretation of ACE encoded labels when they are stored in legacy labels, which is necessary in order for recipient systems to properly detect and decode the label contents.

Note that STD13 octet sequences and ACE data MAY both be provided the same domain name. As such, each STD13 legacy label from a DNS message must be examined and processed independently.

5.2.1. ACE encoded labels

ACE encoded labels always begin with the character sequence of <TBD> (this document uses "zz--" as a placeholder sequence until a formal assignment is made). Any label which contains ACE encoded data MUST begin with this character sequence prefix. Similarly, any label which begins with this character sequence MUST be recognized and processed as an ACE encoded label, according to the rules defined in this specification.

Encoding and encapsulating a label as ACE data is a three-part process, as follows:

a. Encode the canonical UCS character data from the internationalized domain name label into ACE using the procedure defined in <ACE-Z>

b. Preface the encoded output with the "zz--" prefix sequence, thereby indicating that this label contains ACE encoded UCS character data.

c. Determine the length of the encoded data and store this value in the STD13 legacy label's length octet.

Decoding an ACE label is the opposite of that process.

Note that whenever the ACE algorithm encounters a seven-bit character code in the input, it is passed through unmodified to the encoded output. If a label only contains seven-bit character codes, the label MUST NOT be encoded as ACE, and MUST be encoded as either STD13 octet sequences or UTF-8. Forcing a seven-bit label to be encoded as ACE serves no benefit, incurs additional processing on the end-point systems, and can also expose certain security risks. Any system which is capable of generating and deciphering ACE encoded labels is required to treat such sequences as hostile, and MUST dispose of them immediately without any further processing immediately; systems are forbidden to even return these labels in DNS error messages.

Similarly, ACE MUST NOT be used to encode any zero-length labels (including but not specifically limited to the root domain), since the presence of prefix characters in these labels can invalidate their protocol-specific interpretations.

When an STD13 legacy label is received which has "zz--" in the first four character positions, the label MUST be treated as an

ACE-encoded internationalized domain name, and MUST be decoded to its canonical UCS character values for further processing.

Note that STD13 legacy labels MUST be verified before the ACE encoded data is extracted (as per the rules defined in STD13 which govern the STD13 legacy label type), but systems which are compliant with this specification MUST perform all subsequent comparison, caching, or storage operations against the canonical UCS characters, and MUST NOT use the ACE encoded label sequence for any of these operations.

Note that the legacy systems which are not compliant with this specification will treat ACE encoded labels as any other STD13 legacy label.

5.2.2. STD13 octet encoded labels

Any STD13 legacy labels which do not begin with the ACE prefix MUST be treated as STD13 octet encoding sequences. The rules for this process are defined by STD13's default label encapsulation services, although this document also provides some clarifications on the use of this encoding with internationalized domain names and labels.

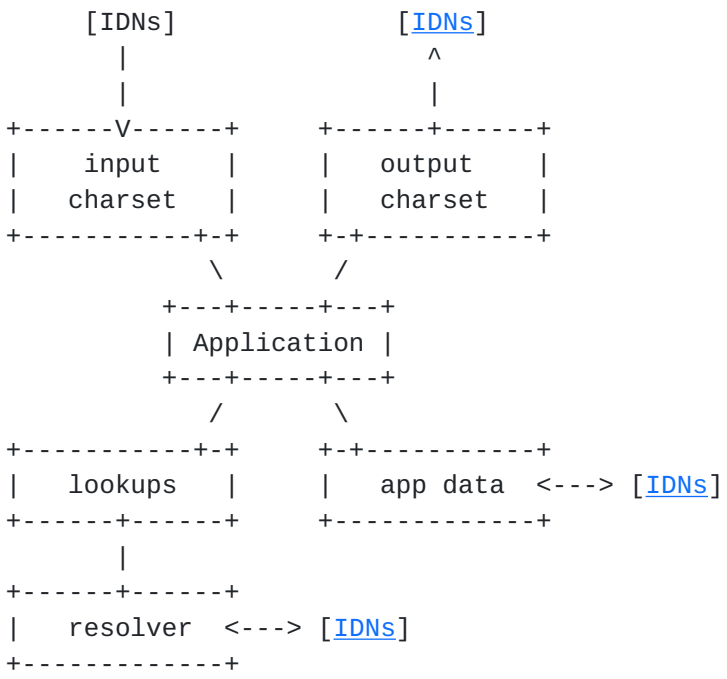
Whenever the STD13 octet sequence is used to encode the labels from an internationalized domain name, the octet values of the canonical UCS characters are stored directly in the label. Because the DNS message is limited to octets, the range of UCS character codes which are eligible for use with STD13 octet sequences is limited to U+0000 through U+00FF. If any UCS character codes outside this range need to be transferred, the internationalized domain name label will have to be encoded as ACE or UTF-8.

Note that comparison operations for the seven-bit range of alphabetic character values MUST be performed in a case-neutral form, although eight-bit code values MUST NOT be normalized or case-converted as part of a comparison operation. These rules are required in order to ensure backwards compatibility with the STD13 compliant systems which may be generating these labels as parts of an STD13 domain name while also supporting the normalization and case-conversion which may have been applied to the UCS characters in the storage or transfer encoding systems.

6. Application Guidelines

As was discussed in section 3.3, there are multiple scenarios in which an application can make use of internationalized domain names, ranging from simple lookups of connection identifiers to abstract encapsulations of unstructured application data. This is an extremely broad range of uses, which is complicated by the extreme pervasiveness of applications and protocols that use domain names for one or more of these purposes.

Furthermore, network applications face a complex array of input and output operations which will cumulatively affect the ability of that application to make use of the internationalized domain name system for various services and functions. These issues are illustrated by the figure below:



As can be seen, the ability for an applications to complete adopt internationalized domain names will be determined by many factors, any one of which could prevent the application from completely incorporating the restrictions and recommendations prescribed by this specification.

In order to allow for a flexible adoption schedule, this specification defines very few mandates that applications must adopt, but instead focuses on recommendations which applications should comply with whenever they need to use internationalized

domain names, and also provides recommendations for situations where the preferred behavior is not feasible. Applications which are compliant with all of the recommendations provided in this specification will be able to generate, store, transfer and resolve internationalized domain names throughout all of their operations, using UTF-8 as a common encoding for all of these operations. Meanwhile, applications which are not in complete compliance with this specification will still be able to make use of the internationalized domain names in these operations, although such access may be limited to using backwards-compatible encodings which require greater amounts of effort to implement and which provide fewer benefits.

6.1. Input and Output Charsets

If an application is unable to accept, process, store or display characters from the complete UCS repertoire, that application's support for internationalized domain names will be somewhat limited, by definition.

Although this document does not mandate any particular charset or encoding which all applications must use for all operations, applications SHOULD use coded character sets or encodings which can handle characters from a reasonable number of scripts.

In particular, the following areas have specific requirements:

* Input charsets and encodings. Since UTF-8 is used as the default encoding for internationalized domain names throughout this specification (and others, such as BCP18), UTF-8 is also RECOMMENDED for use with input encodings of internationalized domain names in particular, although this is not required. Many platforms and development environments support UTF-8 as a local encoding of the UCS and it can be reasonably used with many types of input (such as configuration files), although many systems will require a specific encoding (such as UCS-2, or ISO/IEC 8859-1) in situations which require memory access or keyboard input.

  Regardless of the input encodings used, implementations MUST map domain names and labels to their canonical UCS characters for any normalization and case-conversion work which is subsequently required by any DNS lookups (see section 6.3).

* Output choices will likely be limited to a system-preferred charset or encoding. In general, this document RECOMMENDS that output systems choose an output charset or encoding which reflects the data being provided. However, applications MUST NOT display unknown characters with generic replacement characters (such as boxes or circles) if it is known that the original characters are not available for display with the specified charset, as such characters will almost certainly trigger failure conditions in subsequent protocol operations.

In those situations where adequate input or output charsets or encodings are unavailable, applications MAY use ACE to encode internationalized domain names for the purpose of ensuring that the data is provided intact. Since ACE is capable of representing UCS characters as sequences of seven-bit characters, it is functionally usable as a last line of defense in almost any environment, with the caveat that ACE encoding sequences are extremely cryptic and will likely result in lower levels of usability and functionality.

6.2. Protocol and Application Data

There are several interrelated issues which will determine an application's ability to provide or accept internationalized domain names as protocol or application data, although the principle determining factors for any such usage will generally be the capabilities of the underlying protocol itself.

If a protocol allows negotiation or tagging services in order to distinguish between different encodings, that protocol can likely be extended to support the use of UTF-8 as protocol or application data through command/response negotiation options or through data-type tags. Older protocols which do not provide any negotiation services or which mandate the use of US-ASCII in all data will likely require the use of ACE encoded domain names as a short-term measure until the protocol is made compliant with BCP18.

* Protocol data. If the protocol supports UTF-8 encoded internationalized domain names in commands or responses, then that encoding SHOULD be used wherever it is allowed. If UTF-8 is not supported by the protocol, STD13 octet sequences and/or ACE encoded equivalents of the internationalized domain name MUST be used.

In some cases, this negotiation can be performed on a per-session basis, while in other cases this work will need to be performed for each transaction within the session, while in other cases the internationalized domain names will have to be tagged whenever they are provided as protocol or application data.

The DNS protocol is itself an example of a protocol which requires tagging in order for internationalized domain names to be exchanged within the existing DNS message (with these indicators taking the form of ACE encoding prefixes and EDNS/UTF-8 extended label type codes). Meanwhile, a protocol such as WHOIS can theoretically support a session-wide negotiation option that allowed the use of internationalized domain names as protocol and application data for the duration of that session. Conversely, a protocol such as SMTP will likely require the use of session-specific identifiers for some operations, while other operations may be able to use label tags (similar to the existing support for domain literals, which are identified by a pair of surrounding square brackets).

Regardless of the encodings which are used, implementations MUST map domain names and labels to their canonical UCS characters for any normalization and case-conversion work which is subsequently required as part of a DNS lookup (see section 6.3).

* Structured application data. Structured application data such as URLs and email addresses MUST be processed according to the rules which govern those data formats. Applications MUST NOT perform any conversion or transliteration which is not explicitly prescribed by the governing documents, since non-standard usages are likely to result in misinterpreted data.

* Unstructured application data. Domain names which appear as unstructured data in application content are beyond the control of this specification, and are generally subject to the encoding and formatting desires of the end-users who created the data. Generally speaking, it is RECOMMENDED that applications allow users to enter or view documents in whatever format they prefer, but that any conversion between multiple source and destination charsets and encodings use UCS as the translation intermediary, such

that internationalized domain names are properly converted along with the rest of the application data.

In some cases, the application will need to probe the resolver before it can use internationalized domain names as data. For example, a participating system may need to determine the internationalized domain name of the local system so that it can provide this data in a protocol-specific banner message, and in these cases, the application will have to communicate with the resolver before this data can be provided.

Due to the usage-specific nature of internationalized domain names within protocol and application data streams, each development group will have to analyze the restrictions and capabilities which affect their specific services independently.

6.3. DNS Lookups and Resolver Calls

One of the most frequent uses for domain names is for lookup operations, such as for locating the IP addresses associated with a specified domain name, determining the domain name associated with a specified IP address, or performing a protocol-specific lookup operation for a specific resource record (such as the MX or SOA resource records associated with a specific domain).

Since these lookup operations do not directly affect external protocols or data, internationalized domain names can be used for lookup operations at the application's discretion. For example, applications such as ping and netstat only use domain names for display purposes, and can therefore make immediate use of internationalized domain names within their protocol operations. Similarly, a protocol can be limited to STD13 host identifiers as protocol identifiers which will require the application to provide internationalized domain names as ACE encoded sequences, but any lookup operations which are necessary for the internationalized domain names can still be performed in their native form. In these cases, the protocol operations and lookup operations are separate tasks with separate rules.

Similarly, applications are not required to use internationalized domain names and internationalized resolver APIs for every lookup. In some cases, it may be more efficient for an application to only use internationalized domain names for lookup operations against connection identifiers, and to use STD13 octet sequences or ACE encoded legacy lookups for domain names which were obtained as

protocol or application data (this will be especially true in those cases where the protocol does not yet provide an internationalized domain name data-type). In those cases where an application prefers to use the legacy resolution path, the application MUST use the resolver's legacy APIs. For lookups against internationalized domain names, the application MUST use the resolver's internationalized APIs.

Note that this specification does not define a mandatory encoding which must be used between the applications and the local resolver. However, resolvers MUST provide at least one encoding which is capable of supporting the entire UCS repertoire of character codes, including character codes which are currently unassigned. Since UTF-8 is the default encoding which is used throughout this specification, it is also RECOMMENDED for use with resolver APIs, although this is not required. Resolvers MAY dictate a local encoding, with the only requirement being support for the entire range of UCS character codes.

Regardless of the data being provided or the charset or encoding which is used to provide that data, applications MUST normalize and case-convert any internationalized host identifiers which it generates or receives from a lookup operation. This process MUST use the canonical UCS characters of the domain name according to the rules specified in <nameprep> for every host identifier which is sent to or received from a resolver.

If the application knows that the requested data specifically refers to a host identifier, then the domain name data which is returned by the resolver MUST be normalized and case-converted, and the resulting domain name MUST be compared to the original domain name which was received prior to the normalization and case-conversion steps. If the processed domain name does not match the domain name which was received, the domain name MUST be discarded as malformed.

This step is necessary in order to ensure the integrity and veracity of internationalized domain names which are processed by applications, since there are multiple opportunities for errors to be introduced (such as mistyped entries in the resolver's hosts database, or malicious data which has been purposefully provided in a zone), and these errors can result in sensitive data being directed to the wrong network. Note that the above rule specifically applies to host identifiers and not to all internationalized domain names as a whole; applications MUST NOT arbitrarily normalize and case-convert any and all domain names,

but MUST apply these steps to any and all domain names which are known to be used as host identifiers.

As part of the processing rules for DNS lookups, it is expected that an application can exchange internationalized domain names with the resolver using a charset or encoding which is capable of representing the entire UCS character code range. Towards this objective, applications SHOULD test the capabilities of the resolver prior to transferring internationalized domain names. In those situations where the resolver is unable to support this usage, the application MUST encode the internationalized domain name as STD13 octet sequences or ACE, and pass the resulting STD13 host identifier to the resolver.

7. Resolver Guidelines

Resolvers play a crucial role in the use of internationalized domain names, in that they provide the internationalized namespace which applications work with. As part of this service, resolvers provide encapsulation services for the internationalized domain names which are exchanged with the applications, resolve queries in the internationalized namespace on behalf of the applications, and provide lookup matching for entries which are stored in a local hosts database. Note that resolvers which cache answer data for subsequent operations are also governed by the caching restrictions provided in section 9.

7.1. Resolver APIs

Stub resolvers which communicate directly with applications that are compliant with this specification are strongly encouraged to provide a separate set of APIs for those applications to use whenever internationalized domain names need to be provided in queries or response messages.

The use of an internationalized API will generally facilitate smoother operations for the applications, in that it will allow the application to determine the capabilities of the resolver, to obtain the internationalized domain name of the local system, and to process queries for internationalized domain names as special data types.

Furthermore, the use of internationalized versus legacy APIs provides a way for resolvers to separate internationalized and

legacy application query paths, such that the legacy APIs only result in STD13 legacy labels, while the internationalized APIs generate and trigger EDNS/UTF-8 extended labels. The output formatting of the DNS messages are controlled by tight restrictions, and the use of alternative APIs will likely result in simpler resolver implementations.

For example, it is suggested that applications use the internationalized APIs for all of the DNS lookups they generate, even if the domain name only contains seven-bit characters. This is required in case the queried domain name only exists with a CNAME or PTR resource record which references an internationalized domain name, and the server has to know which encoding to use for that query. If the client had not used the internationalized API for the original lookup of the domain name, the resolver may have chosen the wrong label type, and thus the response data would only be returned as ACE encoded data.

Conversely, older applications which generate malformed eight-bit queries through the legacy APIs will result in those queries being properly rejected by the DNS servers, preventing undue problems with these applications from occurring. For example, an older application may process an internationalized domain name through the system-default charset or encoding (such as MacRoman), which would result in the domain name being malformed when the application tried to do something important with that domain name (such as send an email message over SMTP). The use of multiple APIs causes these malformed applications to break, and the invalid domain names are kept out of the application protocol space.

Internationalized APIs are optional to the extent that an application MAY use an embedded resolver which is known to be capable of generating and processing internationalized domain names through the existing function calls. However, the use of separate APIs for internationalized domain names is encouraged.

Although this document does not mandate any specific APIs, the following functions SHOULD be provided for in some form:

* Test Wide. Applications MUST be able to test the resolver for compliance with this specification. In those cases where this function is performed by some other function (such as one of the following), the capabilities of the resolver MUST be detectable even if the requested operation fails. For example, if an application issues a call for the internationalized domain name of the local system, the

capability of the resolver to handle internationalized domain names MUST be uniquely represented even if the local host name cannot be determined.

* Get Wide X-By-Y. Applications SHOULD be able to specify any resource record associated with any internationalized domain name as part of a lookup operation. Whether this service is provided as a series of lookup-specific APIs or as a general purpose API is up to the resolver.

* Get Wide Local Name. Applications which utilize internationalized domain names as data will need to be able to determine the internationalized form of their local system name for some operations (such as a protocol-specific welcome banner). When this function is called, the resulting data MUST be provided as the canonical UCS character code values, or their equivalent as represented by a locally mandated charset or encoding.

  Note that an ACE equivalent of the system name SHOULD be returned when the relevant legacy API is queried. In those cases where the legacy and internationalized domain names both contain seven-bit character codes (possibly because the host name is only available in US-ASCII, or because the host name was assigned as ACE by an external configuration service), the internationalized host name MUST still be accessible through the internationalized function.

Note that this application does not specify a charset or encoding which must be used by the resolver APIs. However, wherever an internationalized API is presented, the resolver MUST utilize a charset or encoding which supports the entire UCS repertoire of character codes, including character codes which are currently unassigned. Since UTF-8 is the default charset for most of the operations specified in this document, it is also RECOMMENDED for this service, but is not required.

7.2. Query Processing Services

Resolvers which are compliant with the recommendations provided in this specification will provide two query paths, one of which supports STD13 domain names and another which supports internationalized domain names. Technically, there is no requirement for two processing paths, although these paths will

likely exist as conceptual paths even if they are not represented or implemented uniquely in all resolvers.

The legacy processing path is defined by STD13. This document does not update, modify or extend the rules that resolvers operate under when an STD13 compliant domain name is received by a legacy application through any legacy APIs which may exist. However, when an internationalized domain name is received from an internationalized application through any internationalized APIs, the processing rules defined in this section MUST be followed. Note that these rules apply to all resolvers, whether they are stub resolvers, forwarders or caching servers.

Generally speaking, the internationalized domain name resolution process has two major components: processing internationalized domain names as queries, and performing fall-back processing if an EDNS/UTF-8 query is rejected by an authoritative server.

7.2.1. Internationalized queries

Queries for internationalized domain names which are received through internationalized APIs can be expected to have originated at an application which is capable of accepting and processing internationalized domain names in the response messages.

Resolvers MUST encode the labels from the queried domain name as UTF-8 and encapsulate the resulting encoded labels into EDNS/UTF-8 extended labels for transfer within DNS messages, per the instructions provided in section 5.1.

Any and all responses to these queries will also be encoded as UTF-8 and encapsulated in EDNS/UTF-8 extended labels. Resolvers MUST decode the provided response data, convert the labels to their canonical UCS character codes, and return the requested data to the calling application.

The resolver MUST NOT normalize or case convert internationalized domain names which may be received in queries or response messages. Since the queries have originated from applications which have indicated that they are compliant with this specification (via the API) while the responses will have originated from caches or servers which indicate that they are also compliant (via the EDNS/UTF-8 extended labels), those systems are assumed to have normalized and case-converted the domain names before they were generated or stored. Also note that applications

will validate the host identifiers that they receive in response messages, so an additional check is expected to be performed on the answer data by those systems.

7.2.2. Fall-back processing

If a queried server is unable to process EDNS/UTF-8 extended labels, then it is required by STD13 to generate an error signifying the problem. Resolvers MUST interpret these errors, decode the UTF-8 queried domain name, re-encode it as STD13 octets and/or ACE per the instructions provided in section 5.2, and then reissue the query as an STD13 legacy label sequence.

The legacy DNS error responses which will trigger this series of events are FORMERR and NOTIMPL. Any other errors indicate that the EDNS/UTF-8 extended label was successfully processed but that the query was not matched, and those errors MUST be returned to the application. If the fallback processing results in any error responses whatsoever, then the resolver MUST return those errors to the calling application.

Any servers which subsequently receive the fall-back queries and which are compliant with this specification will process the queries as internationalized domain names, and will return the answer data as STD13 octet sequences or ACE encoded data, using the STD13 legacy label.

Generally speaking, fall-back processing serves two purposes:

* Answering the initial query. If a UTF-8 domain name cannot be resolved because a server in the delegation path does not understand the EDNS/UTF-8 label type, the resolver can reissue the query as an ACE encoded legacy label type so that the query proceeds past the problematic server.

* Seeding the resolver's cache. As a result of the above, the resolver will learn about the authoritative name servers for the target zone, and this information can be used for any subsequent queries for domain names within the specified zone (for as long as the data is cached, anyway). As such, any subsequent EDNS/UTF-8 queries which are issued for the portion of the namespace served by that zone will be sent directly to one of those authoritative servers where they can be answered directly. In this regard,

subsequent lookups do not require fall-back processing if they are received during the cache window.

Regardless of whether or not fall-back processing has been performed, if the calling application issued the original query as an internationalized domain name, then the resolver MUST respond to the query in that form as well. This means that the resolver MUST convert any STD13 octet sequences or ACE encoded labels into their canonical UCS characters, convert the answer data into the resolver's native charset or encoding, and return the data to the calling process. The resolver MUST NOT perform any normalization or case-conversion during this process, as such an action can corrupt domain names which are not used for host identifiers.

If the original query was received through the resolver's legacy APIs, then the query MUST be generated and returned in the legacy format, and MUST NOT be converted to an internationalized domain name prior to the query or response being passed through.

Once fall-back processing occurs, the process MUST NOT be repeated for any additional queries in the current lookup operation. No other queries from the current lookup operations MUST NOT be sent as EDNS/UTF-8 extended labels, since multiple fall-back operations can result in time-outs on the client systems.

Because the fall-back process results in two lookups being issued against the rejecting zone, eliminating the fall-back processing as soon as possible will be an operational requirement for many organizations. Any caches or forwarders which are used by stub resolvers within an end-user network are practically required to be able to process the EDNS/UTF-8 queries, since those servers will receive every query which is issued by the stub resolvers. While this isn't a technical requirement (fall-back processing will get around the problematic servers), it will likely prove to be a consideration for network operators looking to support internationalized domain names on their local networks.

This document also strongly encourages the root and TLD servers to be upgraded as soon as possible (even if they do not intend to directly provide UTF-8 domain name delegations), in order to allow those servers to read and process the EDNS/UTF-8 extended labels, thereby reducing the number of fall-back queries which are sent to those servers.

7.3. The Hosts Database

Generally speaking, there are two areas of consideration for stub resolvers that provide local hosts databases for name resolution services. These are the input requirements for internationalized domain names which will be added to the hosts database, and the requirements which govern how queries will be compared to the entries in the hosts database.

Note that resolvers are not required to implement a hosts database or local lookup services (STD3 says "a host MAY also implement a host name translation mechanism that searches a local Internet host table"). However, wherever a hosts database is provided with an internationalized resolver, compliance with the rules specified in this section is required.

If a stub resolver offers the capability to compare internationalized domain names against a local hosts database, that database MUST be compatible with the internationalized domain name rules specified in section 4 of this document.

In particular, the resolver SHOULD allow internationalized domain names with any code values to be stored, even if the canonical UCS characters for those values are undefined or are illegal for use with internationalized host identifiers (this is required to support domain names which are not host identifiers). In those cases where an internationalized domain name specifies an exact sequence of octets for binary comparison, the hosts database MUST provide a mechanism for tagging the eight-bit characters so that they are not interpreted, processed or compared as the canonical UCS character equivalents of those codes.

However, entries which explicitly provide host identifiers MUST be normalized and case-converted prior to being stored. In order to satisfy both of these requirements, it is RECOMMENDED that hosts databases store internationalized host identifiers as untagged data, but that they also provide some sort of tagging service for character code values which are to be returned as-is. STD13 defines an escaping mechanism whereby the decimal value of the octet is prefaced with a reverse-solidus (such as "\193"), which is suggested for this usage.

The storage format of the hosts database MAY use any charset or encoding the resolver deems most suitable for that platform, as long as the rules and restrictions provided above are followed. Since UTF-8 is used as the default encoding throughout this

specification, it is RECOMMENDED as the default encoding for hosts databases as well, although this is not required.

Not all of the applications which use a resolver are likely to be compliant with this specification, so resolvers MUST ensure that they are able to interpret and process any queries from the legacy APIs which provide the ACE equivalent of an internationalized domain name that is stored in the hosts database. When such a query arrives, the domain name MUST be converted to the canonical UCS character codes represented by the ACE encoded sequence and compared to entries in the hosts database in that form (tagged octets excluded). Any internationalized domain names which are required to be returned through the legacy APIs MUST be converted to STD13 octet sequences and/or ACE before they are returned.

8. Server Guidelines

When a zone administrator desires to provide internationalized domain names in a zone, they are presented with two options: they can add the STD13 octets or ACE encoded internationalized domain names to an existing zone, or they can use internationalized zone databases directly. Both of these usage scenarios have their own benefits and restrictions.

Using STD13 octet sequences and ACE with legacy servers allows for the immediate deployment of internationalized domain names on existing servers, and within hierarchies which include internationalized domain names. However, any such queries which originate at applications that are compliant with this specification will always initially fail, guaranteeing that fall-back processing will always occur for those zones.

Conversely, using internationalized zones directly allows servers to process legacy, ACE and EDNS/UTF-8 queries equally, thereby providing greater value to the applications and resolvers which have been made compliant with this specification. However, internationalized zones have additional requirements (most notably, they are required to be upgraded simultaneously), and these will prove burdensome to some zone operators.

This specification focuses on the processing requirements for internationalized zones which support the use of internationalized domain names as explicit data, and which also support the necessary subordinate mechanisms such as EDNS/UTF-8 queries. When STD13 octet sequences or ACE encoded domain names are used with

legacy servers, the rules defined in STD13 for those servers MUST be used.

Note that each zone SHOULD be configurable independently. If a server hosts multiple zones, each of those zones SHOULD be operable as independent entities, with any of them using ACE or internationalized domain names as necessary. This rule is necessary since each zone is likely to have different replication partners and configuration rules which will require different migration strategies.

8.1. Internationalized Zones

All domain names which are published by an internationalized zone MUST be compatible with the restrictions specified in section 4 of this document. In particular, the zone database MUST allow binary domain names to be stored as any octet value, but MUST also comply with the normalization and case-mapping rules when a domain name represents a host identifier. These restrictions MUST be applied as part of the process in which the domain name is being added to the zone database. In those cases where an internationalized domain name specifies an exact sequence of octets for binary comparison, the hosts database MUST provide a mechanism for tagging the eight-bit characters so that they are not interpreted, processed or compared as the canonical UCS character equivalents of those codes. STD13 defines an escaping mechanism whereby the decimal value of the octet is prefaced with a reverse-solidus (such as "\193"), which is suggested for this usage.

Servers which are compliant with this specification MUST be capable of providing UTF-8 and ACE encoded representations of the UCS domain names which are stored in the zone, and servers MUST restrict output to only one label type for any protocol operation, such that queries containing STD13 legacy labels MUST be answered with STD13 octet sequences and/or ACE encoded domain names, while EDNS/UTF-8 queries MUST only be answered with UTF-8 encoded domain names (this not only includes basic operations such as simple queries, but also includes advanced operations such as zone transfers; see section 8.2). Similarly, external operations such as exporting the contents of the zone to a master file (as discussed in section 8.3) MUST result in a single encoding form being used for that specific operation.

Note that the underlying zone database technology which may be employed by any particular server is beyond the scope of this

document. Servers MAY use any database technology, charset or encoding deemed appropriate for the local environment, although the contents of the zone MUST be mapped to the canonical UCS character codes for all comparison operations (octet values excluded). Since UTF-8 is used as the default encoding throughout this specification, it is RECOMMENDED for use as the default encoding with zone databases as well, but is not required.

Servers MUST NOT normalize or case-map any UCS characters which are decoded from UTF-8 or ACE encoded labels, and MUST restrict comparison operations of these labels to precise matches of the UCS domain names which are stored in the zone database. However, the seven bit character codes from any labels which are received as STD13 octet sequences MUST be compared in a case-neutral form, and MUST NOT be normalized as part of the comparison operation.

When a zone is converted to support internationalized domain names, all of the servers which replicate that zone MUST be upgraded. This is required due to ambiguities that can occur with labels which may be encoded as either STD13 octet sequences or ACE data, and where the label only uses character codes from the eight-bit range of character codes (this problem is described in detail in section 4.1.2). In order to ensure that all of the servers for a zone respond to one of those queries correctly, all of the servers which replicate the zone MUST fully support this document and its requirements.

8.2. Namespace Visibility Restrictions

In all cases, the encoding format of the domain names which are returned in response to a query MUST be the same as the encoding format which was used by the query. If the query was provided as a sequence of legacy labels, then all of the domain names which are provided in the response message MUST be provided as legacy labels (containing either ACE or STD13 octet encoded values).

Similarly, if a query is provided as EDNS/UTF-8 encoded data, all domain names which are provided in the response message MUST be provided as UTF-8 encoded data in EDNS/UTF-8 extended labels. In some situations, this process may require the server to perform an extra conversion.

For example, assume that the <idn>.example.com. domain name has two associated MX resource records, one of which points to the UCS domain name of mail.<idn>.example.com, while the other points to

the ACE encoded domain name of mail.<ace>.example.net. (where the "<ace>" label is the ACE equivalent of an internationalized sub-domain in the example.net. zone). If a UTF-8 query arrives for the MX resource records associated with the <idn>.example.com. domain name, both resource records MUST be returned as EDNS/UTF-8 data. In order for this requirement to be satisfied, the server will have to decode the <ace> label to its UCS canonical form for zone storage purposes, and encode the domain name as UTF-8 for transmission whenever an EDNS/UTF-8 answer set is required.

The visibility rules specified in this section are mandatory for every domain name which is provided in any message. If a system requests a zone transfer and uses the EDNS/UTF-8 extended label type in the request, all of the domain names in all of the messages which are sent as part of the zone transfer MUST be provided in their UTF-8 encoded form. Similarly, if a zone transfer is requested and uses the legacy label type, then all of the domain names from all of the messages which are sent as part of the zone transfer MUST be provided as either STD13 octet sequences or ACE encoded data, using the legacy label type.

8.3. The Master File Format

STD13 specifies a "master file" format which is used as a platform-neutral storage and transfer format for importing and exporting the contents of a particular zone. Note that the master file is not the same as the operating database for a zone; the master file format is used (or is useful) for copying a zone to another server, storing a copy of the zone database off-line, emailing a copy of the zone to another user or system, and performing other off-line actions against the database' contents. Once a zone is loaded on a server, however, any database technology can be used for managing the zones and generating response messages.

In order to facilitate the continued use of master files, any zone which is compliant with this specification MUST support the use of UTF-8 as an import and export encoding format for the master file associated with that zone.

Furthermore, compliant versions of a master file are required to have the "$UTF-8" control literal at the beginning of the first line of text in the master file if it contains UTF-8 encoded data. Master files from zones which do not contain UTF-8 encoded domain

names MUST NOT contain the "$UTF-8" control literal in the first print position of any line.

If the master file contains the "$UTF-8" control literal, all of the data within the master file MUST be encoded in UTF-8 as specified by RFC2279, and SHOULD be managed with UTF-8 compliant tools (such as UTF-8 text editors, mailers that support UTF-8 MIME encodings, and so forth).

9. Caching Guidelines

Whenever an internationalized domain name is stored in a cache, it MUST be stored in its canonical UCS character code form, regardless of whether the domain name was received as STD13 octet encoding sequences, UTF-8, or ACE data. Caches MUST NOT normalize or case convert any domain names that they store, as such a process could invalidate domain names that are not used for host identifiers.

Any subsequent queries which are processed through the cache MUST be compared against the stored UCS characters. Internationalized domain name labels which are decoded from UTF-8 or ACE labels MUST NOT be normalized or case-converted as part of the comparison operation, although labels which are provided as STD13 octet sequences MUST be compared as case-neutral octet values.

Caches MUST be capable of providing UTF-8 and ACE encoded representations of the UCS domain names which are stored in the cache, with the appropriate format determined by the format used in the corresponding query. However, answer data MUST be restricted to only one encoding form for any protocol operation, meaning that queries containing legacy labels MUST only be answered with STD13 octet sequences and/or ACE encoded labels, while UTF-8 queries MUST only be answered with UTF-8 encoded domain names.

10. Security Considerations

This document defines an extension to the domain name system, and as such, it inherits the weaknesses which already exist in DNS. Where possible, this specification strengthens DNS with multiple checks. For example, this specification requires that domain names be validated three times before they are used by applications: once on specification, once on entry at the authoritative zone or

hosts database, and once again when the answer data is received by the requesting application. Despite these checks, the root weaknesses inherent in DNS are still present.

This document uses multiple encoding algorithms, although boundary conditions from the existing DNS are preserved for both the source and encoded representations.

11. IANA Considerations

This document requires the use of an EDNS extended label type identification code. This document uses the b000011 ELT code.

12. References

[AMC-ACE-Z] <draft-ietf-idn-amc-ace-z>, "AMC-ACE-Z version 0.3.1"

[NAMEPREP] <draft-ietf-idn-nameprep>, "Preparation of Internationalized Host Names"

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels"

[RFC952] "DoD Internet host table specification"

[STD13] (RFC 1034) "Domain names - concepts and facilities", (RFC 1035) "Domain names - implementation and specification"

[STD3] (RFC 1122) "Requirements for Internet Hosts -- Communication Layers", (RFC1123) "Requirements for Internet Hosts -- Application and Support"

[BCP18] (RFC 2277) "IETF Policy on Character Sets and Languages"

[RFC2279] "UTF-8, a transformation format of ISO 10646"

[RFC2671] "Extension Mechanisms for DNS (EDNS0)"

[ASCII] "ANSI X3.4-1968. USA Standard Code for Information Interchange"

[ISO10646] "ISO/IEC 10646-1:2000. International Standard -- Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane"

13. Acknowledgements

This document is an assembly of multiple ideas and proposals which have been made on the IDN working group mailing list. Many of the ideas presented here have been proposed by multiple parties in one form or another, although Dan Oscarsson is credited for proposing a dual-mode operation which is capable of simultaneously supporting UTF-8 and legacy mode encodings. Other contributors to key elements from this specification (some of them unknowingly or unwillingly) include (alphabetically) Marc Blanchett, Adam Costello, Mark Davis, Martin Duerst, Patrik Faltstrom, Paul Hoffman, David Hopwood, and many others.

14. Editor's Address

Eric A. Hall
ehall@ehsco.com