Domain Name Data-Types

1. Abstract

This document defines syntax and structural rules for a namespace of internationalized domain names, and also clarifies the syntax and structural rules for the existing DNS namespace. Furthermore, this document defines syntax and structural rules for specific types of labels and domain names, and also defines usage rules for specific resource records within the domain name system. This document specifically does not describe any mechanisms for interacting with these namespaces, domain names or resource records, but instead focuses exclusively on the syntax and structural rules.

Table of Contents

2. Introduction

The IDN working group has been developing mechanisms for supporting and interacting with internationalized domain names, although a prerequisite to the completion of any such work is the description of the internationalized namespace itself. During this work, it has also been determined that certain clarifications to the existing DNS namespace are also necessary.

Encodings, protocols and other mechanisms for accessing domain names and resource records within the internationalized namespace are purposefully not described in this document.

Discussion of this document and related work items is currently being held on the "idn@ops.ietf.org" mailing list. To join the list, send a message to <idn-request@ops.ietf.org> with the single word "subscribe" in the body of the message.

Subsequent versions of this draft will be brought to DNSEXT for standards-track development, and will be discussed on the "namedroppers@ops.ietf.org" mailing list. To join that list, send a message to <namedroppers-request@ops.ietf.org> with the single word of "subscribe" in the body of the message.

3. Background and Overview

The Internet (and the ARPANET before it) has had a formal namespace of network resources since RFC 608 [RFC608]. Over the years, however, the syntax rules associated with the global namespace have been changed, with various updates and clarifications being provided in RFC 810 [RFC810], RFC 882 [RFC882], RFC 952 [RFC952], RFC 1034 [RFC1034], RFC 1123 [RFC1123] and RFC 2181 [RFC2181], with each revision expanding upon the namespace syntax to accommodate a more flexible usage model.

The original namespace of network resources defined in [RFC608] used a flat HOSTS.TXT database as a simple list of systems and their network addresses, using a limited subset from the seven-bit US-ASCII charset [ASCII] for the system names. Essentially, the database format was the namespace, with all network services using this one-dimensional namespace for the purpose of specifying systems by name, regardless of whether these hostnames were used for intra-protocol services or for subsequent lookup operations.

The format of the underlying database (and thus the namespace) was redefined by [RFC810] to reflect the coexistence of ARPANET and Internet networks and nodes, redefined again by [RFC952] to allow for multi-label hostnames, and updated in [RFC1123] to slightly expand the allowable character repertoire. Throughout these revisions, the syntax of the namespace was changed somewhat, although it continued to be one-dimensional in nature, reflecting the limitations of the underlying HOSTS.TXT database file.

Once the Domain Name System (DNS) specifications were published (first in [RFC882] and RFC 883 [RFC883], then later in [RFC1034] and RFC 1035 [RFC1035]), the database of network resources and the hostname syntax separated into distinct entities. Although DNS uses an eight-bit syntax internally and allows any of the eight-bit codepoint values to be used for any purposes, most applications and protocols restrict their usage of the namespace to well-known data-types which only use subsets of the available namespace. This has resulted in a distinctly layered namespace, where applications and protocols use the data-type subsets, while the DNS itself uses the full range of characters.

For example, [RFC1034] states that "the old rules for HOSTS.TXT should be followed" for domain names which reference host systems, and most of the application protocols have followed this advice. For all practical purposes, this means that the legacy hostname syntax is implicitly a strong data-type with formal syntax rules, even though it only represents a subset of the global DNS namespace. Meanwhile, a variety of protocol-specific data-types have also been defined for network resources which are not hosts, and these data-types have also been implemented by applications which work with those kinds of resources.

In the end, the DNS namespace essentially exists as two separate layers, with the namespace at large being defined by the underlying DNS service, but with applications and protocols using the data-types and syntax rules which reflect their usage.

Moving forward, the IDN working group has developed an internationalized namespace which uses characters from the Universal Character Set (UCS) [ISO-10646] (a.k.a. Unicode [UNICODE])]. Note that the UCS (and thus the namespace) only defines characters and their logical codepoint values, while external codecs are required to encode the canonical UCS characters into sequences which are suitable for specific environments. As such, the canonical UCS characters cannot be

exchanged in their raw form, but instead can be only exchanged in their encoded form.

Furthermore, there are no characters in the UCS character repertoire for "octet value 0xHH", so the internationalized namespace cannot directly support the eight-bit values used in the DNS namespace. For these reasons, the internationalized and DNS namespaces have to be defined and managed separately, with the internationalized namespace representing logical UCS characters, and with the DNS namespace representing raw and uninterpreted eight-bit values. However, these namespaces can coexist within the class IN database hierarchy as long as the underlying domain names are encoded in a compatible and consistent form. At that point, the only substantive difference between the two namespaces is in the canonical characters which are used by the presentation-layer namespaces at large.

Cumulatively, this means that the internationalized namespace operates at three distinct layers. First of all, applications and protocols have to choose an internationalized data-type which is capable of supporting the characters they need for their domain names, while the protocols also have to choose the encoding formats they will use for the domain names that they exchange. Meanwhile, the end-system applications may need to use another encoding format whenever they map these domain names to the available lookup service(s).

	HOSTS.TXT	DNS Names	IDNs
Application Presentation	database subset	protocol subset	data-type specific (UCS range)
Protocol Transfer	database subset	data-type specific (7- or 8-bit)	protocol encoding (7- or 8-bit)
Subsequent Lookups	HOSTS.TXT subset	8-bit DNS or HOSTS.TXT subset	8-bit DNS or HOSTS.TXT subset

Figure 1: Logical namespace layers and their representations.

The different models which have been described in this section are illustrated in Figure 1 above. As can be seen, the hostname syntax

defined for use with the HOSTS.TXT database essentially provided a monolithic and one-dimensional namespace for applications to use, while the DNS namespace provides multiple data-types for applications and protocols to use, with these data-types representing logical subsets of the underlying eight-bit namespace. Finally, the internationalized namespace also uses logical data-types for the applications and protocols, but also requires that protocols define encoding formats for the domain names they use internally, and for the domain names they pass to the underlying lookup services.

Collectively, there are a variety of different data elements discussed above which require definitions or clarifications. Originally, this document was only meant to provide definitions for the internationalized namespace and its associated data-types, although several issues with the DNS namespace and data-types have also been encountered which require clarifications and definitions of their own. In particular, since the internationalized namespace is unable to use the eight-bit codepoint values from the DNS namespace, the legacy hostname data-type must be defined with an explicit syntax and its usage must be restricted to specific scenarios in order for those domain names to be accessible from the internationalized namespace. Subsequently, this requires that DNS resource records also be redefined to use the data-types which are appropriate to the data they represent, thereby ensuring that host resources in the common hierarchy are always accessible to both namespaces.

On the surface, some of this work may appear to be a reversal of existing standards, since the DNS specifications and the clarifications made in [RFC2181] explicitly allow eight-bit codepoint values to be used with any domain name. In truth, however, this redefinition is a codification of existing practices and recommendations. Specifically, this document encourages applications to define their own data-types and syntax rules when needed but also requires that the common hostname syntax be supported in those places where hosts are specifically referenced, which is essentially a restatement of the [RFC1034] requirements.

The only real difference here is that this document also requires that resource records be explicitly restricted to use the appropriate data-types, rather than being allowed to diverge at will. While this is a reversal of policy as defined in [RFC2181], this is necessary in order to ensure basic interoperability across the different namespaces, and is also necessary in order to

prevent basic interoperability problems from developing due to fragmentation of the class IN hierarchy.

4. The Namespaces

Conceptually, the DNS namespace and the internationalized namespace are separate, in that they allow for different ranges of characters, with the namespaces containing different identifiers. However, both of the namespaces reside in the class IN hierarchy and share a common root in that class, and are effectively the same namespace when the identifiers have been encoded into a compatible and consistent form.

Applications and protocols which specifically utilize any of the data-types defined in this document MUST conform to the syntax rules associated with the parent namespace for that data-type. For example, if an application specifically claims to support the internationalized hostname data-type then that application MUST conform to the requirements associated with the internationalized namespace at large, while applications which claim to conform to the legacy mailbox data-type MUST conform to the requirements of the DNS namespace.

If a protocol supports some other external namespace (such as LDAP directories), then the syntax rules for that protocol SHOULD define specific handling rules which clearly state how that protocol will use each of the namespaces defined here.

4.1. The Class IN Hierarchy

The IN class use a hierarchical structure, where each domain name is represented by a series of labels, and where the entire sequence of labels represents a globally-unique domain name in the hierarchy. Essentially, the IN class hierarchy represents the database portion of the DNS, and therefore represents the storage, transfer and processing services which are used to construct the DNS namespace. Note that the namespace syntax (such as length and character restrictions) is discussed separately in section 4.2. Also note that alternative classes may have their own structural rules which are different from those used in the IN class.

When a domain name from the IN class is used in the DNS, the constituent labels are typically treated as binary sequences (but not always), with each label being prefaced by a length indicator.

The labels are ordered from right-to-left, with the least-specific label at the right edge of the domain name, and with the most-specific labels at the left edge. The right-most label will have a length indicator with the value of zero (it is truly a null label), and represents the root of the class IN hierarchy which is by definition the least-specific label in the hierarchy.

When a domain name is used for lookups, the entire sequence of labels act as a lookup key against a globally-distributed hierarchy of database partitions and their leaf-nodes. Some or all of the labels will identify a specific database partition in the hierarchy, while any remaining labels will identify a particular leaf-node within a partition. As a lookup is processed, the input domain name is matched against the contents of the current partition, with the results of this comparison operation either being a referral to another partition, an answer, or an error. If a referral is returned, the matching process is restarted at the referenced partition, with this process repeating until either an answer or an error is returned.

Domain names from the DNS namespace which are written out in longhand form are usually written as character sequences, with the labels typically being separated by a Full-Stop character (0x2E) from [ASCII]. When used with longhand domain names in the internationalized namespace, the separator mark is either a trailing Full-Stop (U+002E), an Ideographic Full-Stop (U+3002), an Ideographic Full-Width Full-Stop (U+FF0E), or an Ideographic Half-Width Full-Stop (U+FF61) from the UCS. When any of the ideographic forms are used, they MUST be converted to the traditional Full-Stop character when the domain name labels are normalized, and MUST NOT be exchanged with other applications or protocols in their provided form.

Although the separator frequently appears to represent the length indicators in the domain name system, this is not always true. For example, domain names which are written out in longhand form do not typically use a Full-Stop character at the beginning of the domain name to represent the length indicator from the first label, nor do they typically provide a trailing Full-Stop character to represent the root of the hierarchy.

Outside of the domain name system, domain names are typically treated as simple identifiers, with no database context being implied. Humans normally treat domain names as simple identifiers of named network resources, without concern for the leaf-node or partitions which may be referenced. Meanwhile, most applications

and protocols also treat domain names as simple identifiers, although many of them apply syntactical analysis to the domain name before performing any additional processing (even in these situations, however, the domain name will not normally be analyzed for database context).

Note that a one-to-one match between labels, partitions and leaf-nodes is neither required nor implied. Multiple labels may be used to refer to a partition or a leaf-node, as desired. In most cases, the specific database context of a domain name cannot be determined without querying DNS directly.

4.2. The DNS Namespace

These rules represent the allowable syntax of all domain names within the IN class hierarchy as defined by [RFC1034] and [RFC1035]. Alternative classes may define their own namespaces and rules. Subsets of these rules are defined for specific labels and domain names in section 5. The rules provided in this section specifically apply to the DNS namespace at large, and do not define any formal data-types.

4.2.1. Length Restrictions in the DNS Namespace

A label from the DNS namespace is restricted to a minimum of one octet and a maximum of 63 octets, inclusive.

A domain name from the DNS namespace is restricted to a minimum of one octet and a maximum of 255 octets, inclusive. Any number of labels may be provided in a domain name, but the maximum length restriction MUST NOT be exceeded.

Note that many delegation bodies have defined their own minimum length rules for their zones. For example, the historic generic top-level domains (such as com, net and org) require a minimum of two characters for all immediate delegations, while some of the newer generic TLDs have three- and four-character minimums. Since these rules only affect the delegations within those zones (and not the subordinate delegations from the child zones), this usage is not in conflict with any of the other rules defined in this document, and is expressly allowed.

Whenever a domain name is written in longhand form, it SHOULD be restricted to a maximum length which allows a direct conversion to

the DNS format. In particular, a longhand domain name SHOULD allow a length indicator to be added to the first label in the DNS domain name, and SHOULD allow a length indicator to be added to the end of the domain name (representing the root domain), if necessary. In the common case, a longhand domain name will only allow for 253 octets of data so that it can be directly converted to the DNS format.

If a longhand domain name uses the escape syntax described in section 4.2.3, the allowable length of the longhand domain name MAY be extended to accommodate the escape sequence, since a multi-byte escape sequence will generally collapse to a single octet in the DNS message.

Applications and protocols which need to specify the root domain explicitly MUST allow a single Full-Stop character to be specified in the longhand domain name for this purpose. Other application- or protocol-specific syntaxes MAY also be supported for this purpose, if necessary. Note that this usage is not required to be supported unless the application needs to explicitly reference the root domain for some purpose, but since the root domain is not addressable as a host system, this is not a common scenario.

4.2.2. Characters Restrictions in the DNS Namespace

The lower seven-bit range of values (0x00 through 0x7F) from the DNS namespace MUST be interpreted as characters from [ASCII].

The eight-bit range of values (0x80 through 0xFF) are defined in [RFC1034] as opaque octets, with no default character assignments. Therefore, eight-bit values from the DNS namespace MUST NOT be interpreted as any specific charset, characters or encoding, and SHOULD NOT be rendered as such unless the protocol in use has defined a specific data-type which explicitly states otherwise.

When a domain name from the DNS namespace is stored, transferred or compared, the capitalization of the [ASCII] characters in that domain name MUST be preserved as they were provided to the current operation. Secondarily, whenever two domain names from the DNS namespace are compared, the [ASCII] characters in the domain names MUST be treated as case-neutral for the purposes of comparison.

Note that these rules combine such that the capitalization of the input domain name will be preserved across a search operation. For example, the search input of "A.example.COM" MUST match the stored

domain name of "a.EXAMPLE.com", and the capitalization of the input domain name MUST be used for the resulting output. However, if the queried domain name referenced "HOST.c.EXAMPLE.net", that domain name MUST be provided in its original capitalization.

In those scenarios where the input and output domain names are different but they exist in the same branch of the class IN hierarchy, the secondary references to the common domains MUST inherit the capitalization of the input domain name. For example, if the search input of "A.example.COM" matches with "a.EXAMPLE.com" but that domain name only exists as an alias for the domain name of "HOST.b.EXAMPLE.com", then the output domain name MUST be provided as "HOST.b.example.COM", with the capitalization of the input domain name being used to construct the overlapping domain names in the output.

These rules guarantee that the output from the compression algorithm defined in [RFC1035] is always valid. Unless a protocol specifically states otherwise, these rules MUST be followed for all applications and protocols which use domain names and labels from the DNS namespace as specific data.

4.2.3. The DNS Namespace Escape Syntax

Although DNS uses raw eight-bit codepoint values, less than half of the codepoint values have defined character equivalents from [ASCII] which can be rendered, which means that most of the codepoint values cannot be written in a longhand domain name which supports those values.

For example, the octet label data-type supports 256 possible codepoint values (0x00 through 0xFF), while the mailbox label data-type supports all 128 of the seven-bit character codes defined in [ASCII] (0x00 through 0x7F), but only the printable subset of characters from [ASCII] have defined character representations (0x21 through 0x7E). As such, longhand domain names which use these data-types are generally restricted to the printable subset.

Furthermore, some of these domain names make use of characters which are "confusing" to applications and/or their resolvers. For example, many email addresses make use of the Full-Stop character within the local-part element, although this character can easily be misinterpreted as a label separator rather than an embedded Full-Stop character.

[RFC1035] defined a syntax for escaping these characters within the zone database, but it does not require (nor imply) that this mechanism should be supported in other applications, with the result being that most of these applications do not adequately support the necessary syntax. This document corrects this shortcoming by requiring that any application which supports the octet label or mailbox label data-types MUST allow longhand domain names to use the escaping syntax defined herein.

Note that these syntax rules only apply to the octet label and mailbox label data-types. The remaining data-types have tighter character ranges, and do not contain characters which require escaping. Furthermore, applications MUST NOT allow these other data-types to use the escaping syntax whatsoever, as this could result in unexpected characters being inserted into the label or domain name, thereby triggering unexpected failures in other applications or systems.

The escape syntax uses the Reverse-Solidus (0x5C) character as an escape flag, with this flag preceding a printable [ASCII] character or a three-digit decimal value for a specific codepoint value. For example, if a label contains an embedded Full-Stop character, that character may be escaped as either "\." or "\046" (where "46" is the decimal value of the Full-Stop character's codepoint value from [ASCII]).

This escape syntax MUST be used to encapsulate Full-Stop (0x2E), Reverse-Solidus (0x5C), Double-Quote (0x22), a non-printing character from [ASCII] (0x00 through 0x20, or 0x7F) or any of the eight-bit codepoint values (0x80 through 0xFF) whenever one of these domain names is written in longhand form. Protocols which exchange the octet or mailbox data-types as textual data MUST support the use of this escaping syntax within that data. Other application- or protocol-specific syntaxes MAY also be supported for this purpose, if necessary.

However, DNS messages MUST NOT contain the escape sequences, and MUST always use the raw octet value of the escaped character. As such, the escape syntax MUST be interpreted by an application or a resolver (depending on the resolver's capabilities) before these characters are passed into DNS.

4.3. The Internationalized Namespace

These rules represent the allowable syntax of all domain names within the internationalized IN class namespace. Alternative classes may define their own namespaces and rules. Subsets of these rules are defined for specific labels and domain names in section 5. Conversely, the rules provided in this section apply to the internationalized namespace at large, and do not define any formal data-types.

Note that the internationalized namespace is a logical namespace, and does not exist in the same way that the DNS namespace exists. Instead of being a direct mapping to the underlying database, the internationalized namespace is defined as a range of UCS character codes which may be accessed or represented with any of several different encoding mechanisms.

Mechanisms which have been discussed for this purpose include codecs that convert the UCS character codes into seven-bit [ASCII] sequences compatible with the legacy hostname syntax, UCS transfer encodings such as UTF-8, and legacy charsets which can be mapped to the UCS repertoire. Any of these mechanisms (or any others) can be used to represent, store, transfer and compare domain names in the internationalized namespace, as is necessary for the application or protocol at hand.

For example, an internationalized protocol may use UTF-8 domain names as protocol data or arguments, although it may be necessary to convert a domain name into a hostname-compatible encoding whenever a lookup operation is performed. In this scenario, the logical internationalized namespace will be accessed through two different mechanisms, although the canonical domain name will still be represented as the UCS characters. Similarly, conversion between two or more access mechanisms will likely require an intermediate conversion to UCS first, and in this regard, the canonical UCS characters will represent the logical namespace.

Note that this document does not define or describe any codecs or namespace-access mechanisms. However, these different mechanisms have affected the structure and syntax rules of the internationalized namespace at large.

4.3.1. Length Restrictions in the i18n Namespace

For the purposes of defining boundary conditions, a label in the internationalized namespace is restricted to a minimum of one UCS character and a maximum of 63 UCS characters, while a domain name in the internationalized namespace is restricted to a maximum of 255 UCS characters, inclusive. 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 a fewer or greater number of octets, depending on the encoding algorithm in use. For example, an application which uses UCS-4 to represent internationalized domain names will require 32 bits for each UCS character, while an application which uses UTF-8 can require up to 48 bits for each character.

The canonical UCS characters will frequently require conversion into a transfer encoding which will be subject to its own length requirements. For example, the ASCII-compatible encoding mechanism defined in [PUNYCODE] and [IDNA] is subject to the length restrictions inherent in the DNS namespace. Although all encodings have their own requirements, [IDNA] is the only encoding which is known to have hard limits beyond its control. As such, labels in the internationalized namespace MUST be restricted to lengths which can be encoded in their [IDNA] encoded form, with the resulting sequence being limited to the DNS namespace restrictions defined in section 4.2.1. This rule MUST be enforced regardless of any other codecs or access mechanisms which may be available that offer larger sizes.

These rules combine so that an application can restrict the input of a label or domain name (such as in a form) to a certain number of characters, but once the internationalized domain name has been provided and normalized into its canonical form, any subsequent verification of that domain name MUST ensure that the [IDNA] encoded form of that domain name will comply with the DNS namespace limits, which is a maximum of 63 octets for a label, and 255 octets for a domain name.

As with the DNS namespace, domain names written in longhand form MUST leave room for any omitted length indicators when these boundary conditions are tested. In particular, this includes the length indicator at the beginning of the first label and the length indicator at the end of the domain name, both of which are frequently omitted from longhand domain names.

4.3.2. Character Restrictions in the i18n Namespace

The logical internationalized namespace uses the entire UCS, including character codes which are currently unassigned. Currently the UCS occupies a 21-bit range of character code values, containing tens of thousands of assigned characters, and hundreds of thousands of unassigned characters, although this repertoire is expected to grow in size over time.

Internationalized label and domain name data-types MUST declare their own specific ranges of supportable UCS characters, and MUST also define any normalization, case-conversion, and any other transformations which are needed for that data-type. The guidelines and rules for the development of these internationalized domain name data-types are provided in STRINGPREP [STRINGPREP].

In the normal scenario, the data-type rules will result in labels and domain names containing case-specific, strongly-normalized UCS characters. As a result, the internationalized namespace is case-specific, meaning that all storage, transfer, comparison and conversion operations MUST always preserve the capitalization and normalization of the data as it is processed.

Since the domain name provided to the original application can be significantly different from the domain name which is subsequently passed to the underlying protocol, the original domain name MUST NOT be provided to any other applications or protocols if at all possible. Note that certain situations are unavoidable, such as a user copying the hostname from a manually-entered URL into an email message, where the original sequence will not reflect any subsequent normalization. However, this type of bleed-over MUST NOT occur if it can be presented by an application with simple measures.

If a label ONLY contains character codes from the seven-bit [ASCII] range of characters (U+0000 through U+007F), then that label MUST be treated as a legacy label from the DNS namespace for the purposes of comparison. In other words, labels which only contain seven-bit [ASCII] MUST be compared as case-neutral sequences, while all other labels MUST be compared as case-exact sequences. In all situations, the output capitalization MUST reflect the capitalization of the input domain name (since these labels will have been capitalized and normalized according to

their domain name syntax rules, the answer data will also be provided in the appropriate form if this rule is always followed).

5. The DNS Data-Types

As part of the efforts towards strongly defining strict data-types, this document defines syntax rules for different kinds of domain names and labels. Some of the domain name data-types will only contain labels of a single data-type, while some domain name data-types may contain multiple label data-types. For example, a legacy hostname domain name can only contain legacy hostname labels, while an internationalized service locator domain name can contain a mix of different label types.

When one of the internationalized data-types is used with an internationalized protocol, one or more encoding syntaxes MUST be specified by the underlying protocol before the data can be exchanged in a meaningful form. Note that RFC 2277 [RFC2277] states that the preferred encoding for internationalized protocol data is UTF-8.

When one of the internationalized data-types is used with a legacy protocol which only has explicit support for [ASCII], the internationalized data-type MUST be encoded into an ASCII-compatible form before the data can exchanged. The [IDNA] specification describes one such mechanism, and is the preferred encoding form whenever legacy protocols are required to be used.

5.1. Syntax Validation

Each label in a domain name has specific syntax rules which reflect on the data provided in that label. Therefore, applications which validate domain names against a particular data-type SHOULD apply the appropriate syntax rules to each label, as well as validating the domain name in its entirety.

While it may appear that this model is overtly ambiguous, the process of determining the appropriate syntax can be fairly simple if the data-types are consistently enforced. For example, most of the domain name data-types use relatively simple sequences of label data-types, and most applications only support a single domain name data-type for any particular protocol usage. Thus, it is a simple matter to determine if the domain name is valid by comparing it to the syntax rules for the expected usage.

In those cases where the application or protocol allows multiple kinds of domain name to be used, it is still possible to apply some fairly simple lexical analysis to determine the domain name which is in use (a service location label is different from a hostname label, while mailbox labels and octet labels may contain specific escape sequences, and so forth). In those cases where multiple data-types are supported and the domain name data-type cannot be determined (this may happen with an application such as "dig" or "nslookup", for example), then the application MAY choose to simply validate the domain name against the relevant namespace and leave it at that. However, this level of detachment SHOULD NOT be the default behavior; applications SHOULD attempt to validate the labels and domain names to the best of their ability using the available syntax rules.

Specifically, the syntax of a label SHOULD be validated whenever a resource record is added to the replication master for a zone, or whenever an application first creates a domain name for use within that application or its associated network service. These rules are specifically designed to avoid garbage-in, garbage-out syndrome. Subsequent applications and protocol end-points MAY perform syntax validation of any domain names, and specific application protocols MAY require verification by the application end-points, although this will not be required if the participating end-points have performed the necessary validation.

5.2. Defining New Data-Types

Application protocols MAY define any new label or domain name data-types which are needed, although these data-types MUST conform to the rules which govern the controlling namespace, as described in section 4.

Application protocols SHOULD reuse one of the hostname syntaxes if at all possible, since these syntaxes have the widest deployment, and this will facilitate faster adoption of the protocol.

If a protocol needs to support a broader syntax for uses other than referring to hostnames in the DNS or internationalized namespaces, then the protocol SHOULD indicate which operations or external syntaxes require specific exceptions, and document those exception syntaxes separately if possible. For example, if a protocol is capable of using DNS and LDAP equally, this SHOULD be stated explicitly when the protocol-specific data-types are

defined, and a subset data-type which applies specifically to DNS lookups SHOULD also be defined.

5.3. The Root Label and Domain Name

Whenever the DNS message format is used to transport a domain name, [RFC1034] requires the root domain to be specified. However, most applications and protocols do not require or even allow the root domain to be specified as part of the domain names they use internally. In these cases, the applications and protocols will typically leave it up to the local resolver to fully-qualify the domain name as provided, although this process can fail and cause unexpected domain name to be used (see RFC 1535 [RFC1535] for an example and a discussion of this problem).

There are two separate considerations here. First is that an application may need to fully-qualify the domain name in order to prevent misinterpretation. Secondarily, some applications and protocols require that the root domain be provided as the complete domain name, or as part of a domain name (such as a service locator domain name associated with the root zone). The root label is defined to suit both of those purposes, where this is needed. It can be used to explicitly terminate a fully-qualified domain name, and it can be used to explicitly represent the root domain as a standalone entity.

In a multi-label domain name, the root label is represented by a trailing separator mark. The root label MAY be used at the end of any domain name, regardless of its data-type.

In those cases where the root label is provided by itself, the separator mark will specifically represent the root domain of the class IN hierarchy. Note that the root domain is not currently defined as a host (it does not have an IP address), so it cannot currently be used as a connection identifier.

Application protocols SHOULD support the use of a standalone root label as an explicit domain name, although this is specifically not required. Where a protocol defines this usage, it SHOULD NOT be a mandatory requirement for all implementations. Other application- or protocol-specific syntaxes MAY also be supported for this purpose, if necessary. Applications MUST follow the protocol-specific guidelines on this subject.

5.4. The Hostname Labels and Domain Names

Hostnames identify specific systems and zone partitions by name, and are the most widely used of all the data-types. Hostnames are used as the owner name and data values of almost all the common resource records, are used as the connection identifier for almost all protocol-specific syntaxes and generalized applications, and are used for storing system names in local hosts databases, among other purposes.

Since hostnames have such a broad number of uses and potential storage formats, they also have the strictest syntax rules. A hostname is not valid unless each label and the entire domain validate successfully.

5.4.1. Legacy Hostnames

A legacy hostname domain name is a sequence of one or more legacy hostname labels, with an optional root label at the end. Applications which use legacy "hostnames" as specific data-types MUST validate the hostname domain name and label sequences separately and cumulatively.

Note that the syntax for legacy hostnames has undergone many subtle and varied shifts over the years, with multiple updates and revisions allowing for slightly different syntaxes. This document unifies these definitions and clarifies some ambiguities, and is to be considered the definitive reference on the definition of a valid legacy hostname label and domain name.

A legacy hostname label may only contain the Hyphen (0x2D), the numerals "0" through "9" (0x30 through 0x39), the uppercase letters "A" through "Z" (0x41 through 0x5A), and the lowercase letters "a" through "z" (0x61 through 0x7A) from [ASCII]. The first and last character in a hostname label MUST NOT be a Hyphen character, but any other character sequence is valid within the confines of a hostname label.

A legacy hostname domain name is a sequence of one or more legacy hostname labels. However, at least one of the labels MUST contain at least one alphabetic character (a domain name which consists entirely of numeric values has the potential to be confused with an IP address, and this rule prevents this ambiguity). A legacy hostname domain name MAY contain an optional root label at the end

of the domain name, although this usage MUST be explicitly allowed by the application protocol in use.

With the exception of the optional root label at the end of a domain name, a legacy hostname domain name MUST NOT contain any other label data-types. If any of the labels do not validate as legacy hostname labels, or if the entire domain name does not validate as a legacy hostname domain name, then the entire domain name MUST be rejected for use as a legacy hostname domain name.

The length rules associated with the DNS namespace are specifically adopted as the length rules for the legacy hostname label and domain name data-types. This definition updates the definitions provided in [RFC952] and [RFC1123], which had set these lengths at different values. As such, any system which implements a HOSTS.TXT database (or a local equivalent, such as the "/etc/hosts" file on traditional UNIX systems) MUST conform to the length restrictions defined in section 4.2.1.

The syntax rules defined above are somewhat tighter than the syntax allowed in [RFC2181]. However, no standards-track network services have defined hostname syntax rules to use the allowable syntax from [RFC2181]. Instead, almost all application protocols and network services use stricter rules which are highly similar to those defined here.

Applications and protocols which need to support internationalized hostnames MUST use the syntax defined in section 5.4.2.

Applications and protocols which need to support eight-bit octets in their domain names MUST either use the octet label syntax described in section 5.5 or define a new syntax specifically for use with that network service. In either event, new resource records are also likely to be required.

5.4.2. Internationalized Hostnames

An internationalized hostname domain name is a sequence of one or more internationalized hostname labels, with an optional root label at the end. Applications MUST validate the domain name and label sequences separately and cumulatively.

The syntax for internationalized hostname labels is defined in NAMEPREP [NAMEPREP], while this document defines the syntax for internationalized hostname domain names.

The international hostname label rules defined in NAMEPREP require that a label be lowercased and normalized with Unicode normalization form KC prior to use. Due to the massive number of characters which are available for use with internationalized hostname labels, this document cannot summarize the entire set.

Note that an internationalized hostname label which only contains seven-bit codepoint values from the [ASCII] range MUST also validate as a legacy hostname label, using the rules described in section 5.4.1. This is necessary in order for a label to be reusable in both namespaces.

An internationalized hostname domain name is a sequence of internationalized hostname labels, with the additional requirement that at least one of the labels MUST contain a non-numeric character. An internationalized hostname domain name MAY contain an optional root label at the end of the domain name, although this usage MUST be explicitly allowed by the application protocol in use.

With the exception of the optional root label at the end of a domain name, an internationalized hostname domain name MUST NOT contain any other label data-types. If any of the labels do not validate as internationalized hostname labels, or if the entire domain name does not validate as an internationalized hostname domain name, then the entire domain name MUST be rejected for use as an internationalized hostname domain name.

5.5. The Octet Label and Domain Name

The DNS namespace allows labels to contain eight-bit codepoint values, although no standardized representation or interpretation of these values is defined. While [RFC2181] allows these codepoint values to be used with any domain name or label, this document restricts the usage of these values to the octet label data-type.

A octet label essentially provides a direct pass-thru mapping to the underlying DNS namespace. No additional restrictions or interpretations are defined. Multiple octet labels may be used in conjunction with multiple legacy hostname labels (with an optional root label at the end of the end of the domain name) to form an octet domain name.

The UCS character repertoire does not provide any mechanisms for specifying raw octet values, but instead only identifies characters and their codepoint values. As such, eight-bit codepoint values are not accessible to applications which use the internationalized namespace. Instead, those applications will be required to use the DNS namespace directly whenever an octet domain name contains eight-bit codes. However, if the domain name only contains seven-bit characters, then that label can be accessed from the internationalized namespace.

For this reason, applications and protocols SHOULD give preference to the range of characters defined for legacy hostname labels, as this allows the domain name to be accessed from the largest number of sources. However, applications MUST allow the full eight-bit range of values to be specified if the octet domain name data-type is required for the protocol at hand, with the caveat that these labels will be inaccessible from the internationalized namespace.

If a octet label contains a Full-Stop (0x2E), Reverse-Solidus (0x5C), Double-Quote (0x22), a non-printing character from [ASCII] (0x00 through 0x20, or 0x7F) or any of the eight-bit codepoint values (0x80 through 0xFF), then those characters MUST be escaped (using the syntax rules provided in section 4.2.3) whenever the domain name is written in longhand form.

Any number of octet labels may be assigned to a leaf-node in the DNS namespace. However, zone delegations use NS and SOA resource records which use hostname labels, so a fully-qualified domain name outside of the root zone MUST contain at least one legacy hostname label. Since this usage allows for a variable number of octet labels, and since applications outside of the domain name system cannot determine the database context of any given label, this can result in some ambiguity. However, no standards-track network services outside of the DNS currently require the use of octets, so this is fairly narrow area.

5.6. The Mailbox Labels and Domain Names

A variety of resource records make use of mailbox label and domain name data-types in order to encapsulate email addresses into the domain name system (this model allows email addresses to use the DNS compression service). Although there are no known application protocols outside of DNS which use this data-type, the label type still has to be defined for use with DNS, and is therefore defined with its own syntax in this document.

5.6.1. Legacy Mailboxes

The legacy mailbox domain name consists of a single legacy mailbox label followed by one or more legacy hostname labels. In this model, the legacy mailbox label represents the local-part element from an RFC 2822 [RFC2822] email address, while the legacy hostname labels represent the mail domain element from an [RFC2822] email address. When a legacy mailbox domain name is expanded and mapped to an [RFC2822] email address, the legacy mailbox label goes on the left of the "@" separator, while the hostname labels go on the right of the "@" separator.

The local-part element is defined in [RFC2822] as being either a dot-atom or a quoted-string. The dot-atom syntax allows for a relatively complete set of [ASCII] punctuation, numbers and alphabetic characters, while the quoted-string syntax allows for nearly all of the other characters from [ASCII] (certain control characters are globally prohibited in [RFC2822], and these apply to the quoted-string syntax as well).

In order to accommodate as many email addresses as possible, these characters are defined as valid for a legacy mailbox label as well. However, if a legacy mailbox label contains a Full-Stop (0x2E), Reverse-Solidus (0x5C), Double-Quote (0x22), or any non-printing character from [ASCII] (0x00 through 0x20, or 0x7F), then those characters MUST be escaped using the syntax rules provided in section 4.2.3 whenever the domain name is written out in longhand form.

Note that [RFC2821] defines the maximum length of a local-part element as 64 characters, although the maximum length of a legacy label is 63 characters. As a result, not all local-parts can be supported by the legacy mailbox label.

Also note that [RFC2822] also defines the syntax of a "mail domain" as the dot-atom data-type, which allows for a larger subset of [ASCII] characters than the legacy hostname data-type allows. However, [RFC2821] also requires that mail domains be queried through DNS with MX and A resource records, both of which specify host systems. As such, the dot-atom syntax has never been usable with the legacy hostname data-type for the purpose of mail routing. The requirements for stronger data-types and syntax checks defined in this document do not affect this fundamental conflict other than to highlight its presence.

5.6.2. Internationalized Mailboxes

Legacy mailbox labels MAY be used with internationalized hostname labels to form an internationalized mailbox domain name. In this model, the legacy mailbox label represents a legacy local-part, while the internationalized hostname labels represent an international mail domain.

However, note that an internationalized local-part syntax has not yet been defined, and until such a time, an internationalized mailbox label syntax cannot be defined.

5.7. The Service Locator Labels and Domain Names

The service locator label and domain name syntax is used to provide service-specific redirection functions for a particular domain name. This usage is specific to DNS, so it does not have general applicability outside of the domain name system, although the label type still has to be supported, and is therefore defined with its own syntax in this document.

5.7.1. Legacy Service Locators

The service locator label and domain name syntax is defined in RFC 2782 [RFC2782]. In summary, a legacy service locator domain name consists of two legacy service locator labels which uniquely identify a specific network service, with the remainder of the domain name containing hostname and/or root labels.

The service locator labels are identical to the legacy hostname label syntax, with the additional requirement that a service locator label MUST begin with an Underscore (0x5F) character (this usage prevents the SRV resource record's owner domain name from colliding with other owner domain names). In this model, a registered network service is assigned a _service._proto label sequence, with this sequence being appended to the left of a legacy hostname domain name. Application clients can then issue queries for the fully-qualified service locator domain name, with the resulting answer data providing indicating which hosts offer that service on behalf of the queried domain name.

Note that zone delegation requires the use of legacy hostname labels, so a fully-qualified domain name for an SRV resource record associated with a domain name outside of the root zone MUST contain at least one legacy hostname label. However, an application can also query for an SRV resource record associated with the root domain itself, and in that scenario, the fully-qualified domain name would be "_service._proto.", with the trailing separator mark explicitly representing the root domain.

5.7.2. Internationalized Service Locators

Service locator labels MAY be used with internationalized hostname labels. In this model, the legacy service locator labels represent a known service associated with an international domain.

Note that [RFC2277] says that protocol identifiers do not need to be internationalized. As such, there is no requirement foreseen to allow non-ASCII characters in the service locator label syntax.

6. Resource Records and Query Types

This section describes the domain name data-types in use with all of the registered resource records and query-types. These definitions include the valid owner domain names for a particular kind of resource record, and also include the valid domain names for the resource record data sections. Note that each of these rules only use domain name data-types, while those data-types are defined by constituent sets of label data-types.

6.1. Resource Records

Resource records may be provided in any section of a DNS message. When they are provided in the question section as the query question, they only have owner names. When they are provided in any other section, they have owner names and resource record data. All resource records have owner name syntax rules, while those resource records which also provide domain names in resource record data also have syntax rules for those domain names.

All new resource records MUST be defined with syntax rules appropriate to that resource record.

Resource Record Name: IPv4 Address
Resource Record Mnemonic: A
Resource Record Code: 1
Defined In: [RFC1035]
Owner Name: Hostname
Resource Record Data: (no domain name data-types)

Resource Record Name: Name Server
Resource Record Mnemonic: NS
Resource Record Code: 2
Defined In: [RFC1035]
Owner Name: Hostname (delegated zone partition)
Resource Record Data: Hostname (authoritative server)
Note: All domain delegations MUST use the hostname data-type.

Resource Record Name: Mail Destination
Resource Record Mnemonic: MD
Resource Record Code: 3
Defined In: [RFC1035]
Owner Name: Hostname (mail domain)
Resource Record Data: Hostname (delivery server)
Note: Obsoleted and deprecated by RFC1035 in favor of MX

Resource Record Name: Mail Forwarder
Resource Record Mnemonic: MF
Resource Record Code: 4
Defined In: [RFC1035]
Owner Name: Hostname (mail domain)
Resource Record Data: Hostname (relay server)
Note: Obsoleted and deprecated by RFC1035 in favor of MX

Resource Record Name: Canonical Name
Resource Record Mnemonic: CNAME
Resource Record Code: 5
Defined In: [RFC1035]
Owner Name: inherited from target owner name
Resource Record Data: inherited from target owner name
Note: The owner domain name and resource record data are both
inherited from the target of the CNAME resource record. For
example, if a CNAME resource record references an A resource
record, then the owner name and the resource record data
both use the Hostname domain name data-type. However, if a
CNAME resource record references an SRV resource record,
then the owner name and the resource record data both use
the Service Locator domain name data-type.

```
   Resource Record Name: Start-of-Authority
   Resource Record Mnemonic: SOA
   Resource Record Code: 6
   Defined In: [RFC1035]
   Owner Name: Hostname
   Resource Record Data: multiple fields (see below)
         MNAME:          Hostname (replication master server)
         RNAME:          Mailbox (administrator's email address)
         SERIAL:        (no domain name data-types)
         REFRESH:       (no domain name data-types)
         RETRY:          (no domain name data-types)
         EXPIRE:                (no domain name data-types)
         MINIMUM:       (no domain name data-types)

   Resource Record Name: Mailbox
   Resource Record Mnemonic: MB
   Resource Record Code: 7
   Defined In: [RFC1035]
   Owner Name: Mailbox (recipient email address)
   Resource Record Data: Hostname (delivery server)

   Resource Record Name: Mail Group
   Resource Record Mnemonic: MG
   Resource Record Code: 8
   Defined In: [RFC1035]
   Owner Name: Mailbox (original email address)
   Resource Record Data: Mailbox (expanded email address)

   Resource Record Name: Mail Rename
   Resource Record Mnemonic: MR
   Resource Record Code: 9
   Defined In: [RFC1035]
   Owner Name: Mailbox (original email address)
   Resource Record Data: Mailbox (new email address)

   Resource Record Name: Null
   Resource Record Mnemonic: NULL
   Resource Record Code: 10
   Defined In: [RFC1035]
   Owner Name: Legacy Octets
   Resource Record Data: (no domain name data-types)
```

Resource Record Name: Well-Known Services
Resource Record Mnemonic: WKS
Resource Record Code: 11
Defined In: [RFC1035]
Owner Name: Hostname
Resource Record Data: (no domain name data-types)

Resource Record Name: Pointer
Resource Record Mnemonic: PTR
Resource Record Code: 12
Defined In: [RFC1035]
Owner Name: inherited from target owner name
Resource Record Data: inherited from target owner name
Note: Although PTR resource records are most often used to provide
reverse-lookup mappings, the data can be used for any domain
name which needs to point to another domain name. As such,
the owner name and the resource record data must both
inherit the domain name data-type in use with the
destination.

Resource Record Name: Host Information
Resource Record Mnemonic: HINFO
Resource Record Code: 13
Defined In: [RFC1035]
Owner Name: Hostname
Resource Record Data: (no domain name data-types)

Resource Record Name: Mail List Information
Resource Record Mnemonic: MINFO
Resource Record Code: 14
Defined In: [RFC1035]
Owner Name: Mailbox (mailing list primary address)
Resource Record Data: multiple fields (see below)
RMAILBOX: Mailbox / Root (responsible party address)
EMAILBOX: Mailbox / Root (error-handler mailbox)
Note: MINFO defines an application-specific interpretation for the
root domain in the resource record as an alternative to the
Mailbox data-type.

```
   Resource Record Name: Mail Exchange
   Resource Record Mnemonic: MX
   Resource Record Code: 15
   Defined In: [RFC1035]
   Owner Name: Hostname (mail domain)
   Resource Record Data: multiple fields (see below)
         PREFERENCE:   (no domain name data-types)
         DESTINATION:  Hostname (mail server)

   Resource Record Name: Text
   Resource Record Mnemonic: TXT
   Resource Record Code: 16
   Defined In: [RFC1035]
   Owner Name: Legacy Octets
   Resource Record Data: (no domain name data-types)
   Note: The TXT resource record is commonly used as a proving ground
         for new resource records, and this must continue to be
         supported.

   Resource Record Name: Responsible Person
   Resource Record Mnemonic: RP
   Resource Record Code: 17
   Defined In: RFC 1183 [RFC1183]
   Owner Name: Hostname
   Resource Record Data: multiple fields (see below)
         RMAILBOX:     Mailbox (responsible person contact address)
         DETAILS:      Legacy Octets (pointer to TXT record)
   Note: Binding this resource record to the hostname data-type may
         artificially limits its usefulness, although it results in
         greater predictability and consistency in the
         internationalized namespace.

   Resource Record Name: AFS Database Entry
   Resource Record Mnemonic: AFSDB
   Resource Record Code: 18
   Defined In: [RFC1183]
   Owner Name: Hostname
   Resource Record Data: multiple fields (see below)
         PREFERENCE:   (no domain name data-types)
         DESTINATION:  Hostname (AFS server)
```

```
   Resource Record Name: X.25 Number
   Resource Record Mnemonic: X.25
   Resource Record Code: 19
   Defined In: [RFC1183]
   Owner Name: Hostname (host)
   Resource Record Data: (no domain name data-types)

   Resource Record Name: ISDN Number
   Resource Record Mnemonic: ISDN
   Resource Record Code: 20
   Defined In: [RFC1183]
   Owner Name: Hostname (host)
   Resource Record Data: (no domain name data-types)

   Resource Record Name: Route-Through
   Resource Record Mnemonic: RT
   Resource Record Code: 21
   Defined In: [RFC1183]
   Owner Name: Hostname (host)
   Resource Record Data: multiple fields (see below)
         PREFERENCE:   (no domain name data-types)
         DESTINATION:  Hostname (next-hop host)

   Resource Record Name: OSI NSAP Address
   Resource Record Mnemonic: NSAP
   Resource Record Code: 22
   Defined In: RFC 1706 [RFC1706]
   Owner Name: Hostname
   Resource Record Data: (no domain name data-types)

   Resource Record Name: OSI NSAP Pointer
   Resource Record Mnemonic: NSAP-PTR
   Resource Record Code: 23
   Defined In: [RFC1706]
   Owner Name: Hostname (hexadecimal NSAP address)
   Resource Record Data: Hostname (host)

   Resource Record Name: Signature
   Resource Record Mnemonic: SIG
   Resource Record Code: 24
   Defined In: RFC 2535 [RFC2535] and RFC 2931 [RFC2931]
   Owner Name: Hostname (?)
   Resource Record Data: (no domain name data-types)
```

Resource Record Name: Public Key
Resource Record Mnemonic: KEY
Resource Record Code: 25
Defined In: [RFC2535]
Owner Name: Legacy Octets
Resource Record Data: (no domain name data-types)
Note: The owner name must be treated as unstructured, since the
KEY resource record may be bound to any domain name.

Resource Record Name: Pointer to X.400 Mapping
Resource Record Mnemonic: PX
Resource Record Code: 26
Defined In: RFC 2163 [RFC2163]
Owner Name: Hostname (encoded X.400 mail domain)
Resource Record Data: multiple fields (see below)
RFC822-MAIL: Hostname (mail domain)
X400-MAIL: Hostname (mail domain)

Resource Record Name: Geographical Position
Resource Record Mnemonic: GPOS
Resource Record Code: 27
Defined In: RFC 1712 [RFC1712]
Owner Name: Hostname
Resource Record Data: (no domain name data-types)

Resource Record Name: IPv6 Simple Address
Resource Record Mnemonic: AAAA
Resource Record Code: 28
Defined In: RFC 1886 [RFC1886]
Owner Name: Hostname
Resource Record Data: (no domain name data-types)

Resource Record Name: Location
Resource Record Mnemonic: LOC
Resource Record Code: 29
Defined In: RFC 1876 [RFC1876]
Owner Name: Hostname
Resource Record Data: (no domain name data-types)

```
   Resource Record Name: Next Record
   Resource Record Mnemonic: NXT
   Resource Record Code: 30
   Defined In: [RFC2535]
   Owner Name: Legacy Octets
   Resource Record Data: multiple fields (see below)
         NEXT-OWNER:   Legacy Octets (the next domain name)
         TYPE:         (no domain name data-types)
   Note: The owner name must be treated as unstructured, since the
         NXT resource record may be bound to any domain name.

   Resource Record Name: Service Locator
   Resource Record Mnemonic: SRV
   Resource Record Code: 33
   Defined In: [RFC2782]
   Owner Name: Service Locator
   Resource Record Data: multiple fields (see below)
         PRIORITY:     (no domain name data-types)
         WEIGHT:               (no domain name data-types)
         PORT:         (no domain name data-types)
         TARGET:               Hostname (target server)

   Resource Record Name: ATM Address
   Resource Record Mnemonic: ATMA
   Resource Record Code: 34
   Defined In: N/A (see ATM Forum standards)
   Owner Name: Hostname
   Resource Record Data: (no domain name data-types)

   Resource Record Name: Naming Authority Pointer
   Resource Record Mnemonic: NAPTR
   Resource Record Code: 35
   Defined In: RFC 2915 [RFC2915]
   Owner Name: Hostname
   Resource Record Data: multiple fields (see below)
         ORDER:         (no domain name data-types)
         PREFERENCE:   (no domain name data-types)
         FLAGS:         (no domain name data-types)
         SERVICE:      (no domain name data-types)
         REGEXPS:      (no domain name data-types)
         REPLACEMENT:  Hostname / Service Locator (see notes)
   Note: The domain name provided in the REPLACEMENT sub-field can
         reference a NAPTR, SRV or A resource record by its owner
         name, depending on the value of the FLAGS sub-field.
```

Resource Record Name: Key Exchange
Resource Record Mnemonic: KX
Resource Record Code: 36
Defined In: RFC 2230 [RFC2230]
Owner Name: Legacy Octets (any domain name)
Resource Record Data: multiple fields (see below)
PREFERENCE: (no domain name data-types)
DESTINATION: Hostname (key server)
Note: The owner name must be treated as unstructured, since the KX resource record may be bound to any domain name.

Resource Record Name: Certificate
Resource Record Mnemonic: CERT
Resource Record Code: 37
Defined In: RFC 2538 [RFC2538]
Owner Name: Hostname
Resource Record Data: (no domain name data-types)
Note: The owner name must be treated as unstructured, since the CERT resource record may be bound to any domain name.

Resource Record Name: IPv6 Complex Address
Resource Record Mnemonic: A6
Resource Record Code: 38
Defined In: RFC 2874 [RFC2874]
Owner Name: Hostname (host)
Resource Record Data: (no domain name data-types)

Resource Record Name: Domain Name Redirection
Resource Record Mnemonic: DNAME
Resource Record Code: 39
Defined In: RFC 2672 [RFC2672]
Owner Name: Hostname
Resource Record Data: Hostname

Resource Record Name: Extended Option
Resource Record Mnemonic: OPT
Resource Record Code: 41
Defined In: RFC 2671 [RFC2671]
Owner Name: Root
Resource Record Data: (no domain name data-types)

Resource Record Name: Transaction Key
Resource Record Mnemonic: TKEY
Resource Record Code: 249
Defined In: RFC 2930 [RFC2930]
Owner Name: Legacy Octets
Resource Record Data: (no domain name data-types)
Note: The owner name must be treated as unstructured, since the
TKEY resource record may be bound to any domain name.

Resource Record Name: Transaction Signature
Resource Record Mnemonic: TSIG
Resource Record Code: 250
Defined In: RFC 2845 [RFC2845]
Owner Name: Legacy Octets
Resource Record Data: (no domain name data-types)
Note: The owner name must be treated as unstructured, since the
TSIG resource record may be bound to any domain name.

6.2. Query Types

Apart from the resource records defined in section 6.1 above, there are also a handful of query types. Query types are only provided in the question section of a DNS message, and do not have resource record data. However, their owner names have domain name data-types which require standardization.

All new query-types MUST be defined with syntax rules appropriate to that query-type.

Query Name: Incremental Transfer
Query Mnemonic: IXFR
Query Code: 251
Defined In: RFC 1995 [RFC1995]
Owner Name: Hostname

Query Name: Zone Transfer
Query Mnemonic: AXFR
Query Code: 252
Defined In: [RFC1035]
Owner Name: Hostname

Query Name: Mailbox Records
Query Mnemonic: MAILB
Query Code: 253
Defined In: [RFC1035]
Owner Name: Mailbox
Note: This query-type requests all of the Mailbox, Mail Group, Mail Rename and Mail List resource records associated with an email address.

Query Name: Mail Transfer Records
Query Mnemonic: MAILA
Query Code: 254
Defined In: [RFC1035]
Owner Name: Hostname
Note: Obsoleted and deprecated by RFC1035 in favor of MX, but used to request all of the Mail Forwarder and Mail Destination resource records associated with a mail domain.

Query Name: All Records
Query Mnemonic: "*" or ALL
Query Code: 255
Defined In: [RFC1035]
Owner Name: Hostname

7. Security Considerations

This document does not change any on-the-wire formats, and therefore does not introduce any new security risks within the affected protocols. However, it is the author's hope that by defining strict syntaxes for domain names and labels that overall security can be improved as a result of higher predictability and better development practices.

8. IANA Considerations

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

9. References

[RFC608] RFC 608, "HOST NAMES ON-LINE", M. Kudlick, January 1974.

[RFC810] RFC 810, "DoD INTERNET HOST TABLE SPECIFICATION", E. Feinler et al, March 1982.

[RFC882] RFC 882, "DOMAIN NAMES - CONCEPTS and FACILITIES", P. Mockapetris, November 1983.

[RFC952] RFC 952, "DOD INTERNET HOST TABLE SPECIFICATION", K. Harrenstien et al, October 1985.

[RFC1034] RFC 1034, "DOMAIN NAMES - CONCEPTS and FACILITIES", P. Mockapetris, November 1987.

[RFC1123] RFC 1123, "Requirements for Internet Hosts -- Application and Support", R. Braden, October 1989.

[RFC2181] RFC 2181, "Clarifications to the DNS Specification", R. Elz et al, July 1997.

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

[RFC883] RFC 883, "DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION", P. Mockapetris, November 1983.

[RFC1035] RFC 1035, "DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION", P. Mockapetris, November 1987.

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

[UNICODE] "The Unicode Consortium, The Unicode Standard, Version 3.0", Addison-Wesley: Reading, MA, 2000. Update to version 3.1, 2001. Update to version 3.2, 2002.

[PUNYCODE] Internet-Draft, "Punycode:An encoding of Unicode for use with IDNA", A. Costello, May 2002.

[IDNA] Internet-Draft, "Internationalizing Domain Names In Applications (IDNA)", P. Faltstrom et al, May 2002.

[STRINGPREP] Internet-Draft, "Preparation of Internationalized Strings", P. Hoffman et al, May 2002.

[NAMEPREP] Internet-Draft, "Nameprep: A Stringprep Profile for Internationalized Domain Names", P. Hoffman et al, May 2002.

[RFC1535] RFC 1535, "A Security Problem and Proposed Correction With Widely Deployed DNS Software", E. Gavron, October 1993.

[RFC2821] RFC 2821, "Simple Mail Transfer Protocol", J. Klensin, April 2001.

[RFC2822] RFC 2822, "Internet Message Format", P. Resnick, April 2001.

[RFC2782] RFC 2782, "A DNS RR for specifying the location of services (DNS SRV)", A. Gulbrandsen et al, February 2000.

[RFC2277] RFC 2277, "IETF Policy on Character Sets and Languages", H. Alvestrand, January 1998.

[RFC1183] RFC 1183, "New DNS RR Definitions", C. Everhart et al, October 1990.

[RFC1706] RFC 1706, "DNS NSAP Resource Records", B. Manning et al, October 1994.

[RFC2535] RFC 2535, "Domain Name System Security Extensions", D. Eastlake, March 1999.

[RFC2931] RFC 2931, "DNS Request and Transaction Signatures (SIG(0)s)", D. Eastlake, September 2000.

[RFC2163] RFC 2163, "Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM)", C. Allocchio, January 1998.

[RFC1712] RFC 1712, "DNS Encoding of Geographical Location", C. Farrell et al, November 1994.

[RFC1886] RFC 1886, "DNS Extensions to support IP version 6", S. Thomson et al, December 1995.

[RFC1876] RFC 1876, "A Means for Expressing Location Information in the Domain Name System", C. Davis et al, January 1996.

[RFC2915] RFC 2915, "The Naming Authority Pointer (NAPTR) DNS Resource Record", M. Mealling et al, September 2000.

[RFC2230] RFC 2230, "Key Exchange Delegation Record for the DNS", R. Atkinson, November 1997.

[RFC2538] RFC 2538, "Storing Certificates in the Domain Name System (DNS)", D. Eastlake et al, March 1999.

[RFC2874] RFC 2874, "DNS Extensions to Support IPv6 Address Aggregation and Renumbering", M. Crawford et al, July 2000.

[RFC2672] RFC 2672, "Non-Terminal DNS Name Redirection", M. Crawford, August 1999.

[RFC2671] RFC 2671, "Extension Mechanisms for DNS (EDNS0)", P. Vixie, August 1999.

[RFC2930] RFC 2930, "Secret Key Establishment for DNS (TKEY RR)", D. Eastlake, September 2000.

[RFC2845] RFC 2845, "Secret Key Transaction Authentication for DNS (TSIG)", P. Vixie et al, May 2000.

[RFC1995] RFC 1995, "Incremental Zone Transfer in DNS", M. Ohta, August 1996.

10. Acknowledgements

The author made multiple attempts at avoiding this work. David Hopwood and Mark Andrews are credited with arguing that it needed to be done, and John Klensin is credited with providing helpful feedback on how it should be done.

11. Author's Address

Eric A. Hall
ehall@ehsco.com