Internet Draft
draft-ietf-idn-udns-03.txt
Updates: RFC 2181, 1035, 1034, 2535
Expires: 19 February 2002

Dan Oscarsson
Telia ProSoft
19 August 2001

Using the Universal Character Set in the Domain Name System (UDNS)

Status of this memo

Abstract

Since the Domain Name System (DNS) [RFC1035] was created there have been a desire to use other characters than ASCII in domain names. Lately this desire have grown very strong and several groups have started to experiment with non-ASCII names. This document defines how the Universal Character Set (UCS) [ISO10646] is to be used in DNS. It includes both a transition scheme for older software supporting non-ASCII handling in applications only, as well as how to use UCS in labels and having more than 63 octets in a label.

1. Introduction

While the need for non-ASCII domain names have existed since the creation of the DNS, the need have increased very much during the last few years. Currently there are at least two implementations using UTF-8 in use, and others using other methods.

To avoid several different implementations of non-ASCII names in DNS that do not work together, and to avoid breaking the current ASCII only DNS, there is an immediate need to standardise how DNS shall handle non-ASCII names.

While the DNS protocol allow any octet in character data, so far the octets are only defined for the ASCII code points. Octets outside the ASCII range have no defined interpretation. This document defines how all octets are to be used in character data allowing a standardised way to use non-ASCII in DNS.

The specification here conforms to the IDN requirements [IDNREQ].

1.1 Terminology

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 [RFC2119].

IDN: Internationalised Domain Name, here used to mean a domain name containing non-ASCII characters.

ACE: ASCII Compatible Encoding. Used to encode IDNs in a way compatible with the ASCII host name syntax.

1.2 Previous versions of this document

This version contains just minor corrections to the 4:th version.

The third version of this document included a way to return both ASCII and non-ASCII versions of a name. As this could not be guaranteed to work it has been removed.

The second version of this document was available as draft-ietf-idn-udns-00.txt. It included a lot of possibilities as well as a flag bit that is now removed.

The first version of this document was available as draft-oscarsson-i18ndns-00.txt.

2. The DNS Protocol

The DNS protocol is used when communicating between DNS servers and other DNS servers or DNS clients. User interface issues like the format of zone files or how to enter or display domain names are not part of the protocol.

The update of the protocol defined here can be used immediately as it

is fully compatible with the DNS of today.

For a long time there will be software understanding UCS in DNS and software only understanding ASCII in DNS. It is therefore necessary to support a mixing of both types. For the following text software understanding UCS in DNS will be called UDNS aware.

This specification supports the following scenarios:

- UDNS unaware client, UDNS aware DNS server
- UDNS aware client, UDNS unaware DNS server
- UDNS aware client, UDNS aware DNS server

2.1 Fundamentals

2.1.1 Standard Character Encoding (SCE)

Character data need to be able to represent as much as possible of the characters in the world as well as being compatible with ASCII. Character data is used in labels and in text fields in the RDATA part of a RR.

The Standard Character Encoding of character data used in the DNS protocol MUST:
- Use ISO 10646 (UCS) [ISO10646] as coded character set.
- Be normalised using form C as defined in Unicode technical report #15 [UTR15]. See also [CHNORM].
- Encoded using the UTF-8 [RFC2279] character encoding scheme.

2.1.2 Binary Comparison Format (BCF)

RFC 1035 states that the labels of a name are matched case-insensitively. When using UCS this is no longer enough as there are other forms than case that need to match as equivalent. Form-insensitive matching of UCS includes:
- Letters of different case are compared as the same character.
- Code points of primary typographical variations of the same character are compared as the same character. An example is double width/normal width characters or presentation forms of a character.
- Some characters are represented with multiple code points in UCS. All code points of one character must compare as the same. For example the degree Kelvin sign is the same as the letter K.

The original definition is now extended to be: labels must be compared using form-insensitivity.

To handle form-insensitivity it is here defined the Binary Comparison Format (BCF) to which strings can be mapped. After strings is mapped to BCF they can be compared using binary string comparison. Implementors may implement the form-insensitive comparison without using BCF, as long as the results are the same.

Mapping of a label to BCF is typically done by steps like: changing all upper case letters to lower case, mapping different forms to one form and changing different code points of one character into a single code point.

For the UCS character code range 0-255 (ASCII and ISO 8859-1) the BCF MUST be done by mapping all upper case characters to lower case following the one to one mapping as defined in the Unicode 3.0 Character Database [UDATA].

The definition of the Binary Comparison Format (BCF) for the rest of UCS will be defined in a separate document. The nearest today is [NAMEPREP].

2.1.3 Backward Compatibility Encoding (BCE)

To support older software expecting only ASCII and to support downgrading from 8-bit to 7-bit ASCII in other protocols (like SMTP) a Backward Compatibility Encoding (BCE) is available. It is a transition mechanism and will no longer be supported at some future time when it is so decided.

The Backward Compatibility Encoding (BCE) of a label is defined as the BCF of the label encoded using an ASCII Compatible Encoding (ACE).

The definition of the ACE to be used, is defined in a separate document. Typical definitions that are suitable are [SACE] and [RACE].

The reason that the BCF form of the label is used is to support solutions where only applications know about non-ASCII labels. By using BCF the server need not know about UCS and can just do binary matching so it can be handled in old servers. Though due to the fact that BCF destroys information contained in the original form of a label it is impossible to return the original form to a client using BCE.

2.1.4 Long names

The current DNS protocol limits a label to 63 octets. As UTF-8 take more than one octet for some characters, an UTF-8 name cannot have 63

characters in a label like an ASCII name can. For example a name using Hangul would have a maximum of 21 characters.

The limits imposed by RFC 1035 is 63 octets per label and 255 octets for the full name. The 255 limit is not a protocol limit but one to simplify implementations.

To support longer names a long label type is defined using [RFC2671] as extended label 0b000011 (the label type will be assigned by IANA and may not be the number used here).

```
                    1 1 1 1 1 1 1 1 1 1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|0 1 0 0 0 0 1 1|  length       |  label data ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

length: length of label in octets
label data: the label

The long label MUST be handled by all software following this specification. Also, they MUST support a UDP packet size of up to 1280 bytes.

The limits for labels are updated since RFC 1025 as follows:
A label is limited to a maximum of 63 character code points in UCS normalised using Unicode form C. The full name is limited to a maximum of 255 character code points normalised as for a label.

A long label MUST always use the Standard Character Encoding (SCE).

As long labels are not understood by older software, a response MUST not include a long label unless the query did. At a later date, IETF may change this.

2.2 Rules for matching of domain names in UDNS aware DNS servers

To be able to handle correct domain name matching in lookups, the following MUST be followed by DNS servers:

- Do matching on authorative data using form-insensitive matching for the characters used in the data (for example a zone using only ASCII need only handle matching of ASCII characters).
- On non-authorative data, either do binary matching or case-insensitive matching on ASCII letters and binary matching on all others.

The effect of the above is:

- only servers handling authorative data must implement form-insensitive matching of names. And they need only implement the subset needed for the subset of characters of UCS they support in their authorative zones.
- it normally gives fast lookup because data is usually sent like: resolver <-> server <-> authorative server.
  While form-insensitive matching can be complex and CPU consuming, the server in the middle will do caching with only simple and fast binary matching. So the impact of complex matching rules should not slow down DNS very much.

2.3 Mixing of UDNS aware and non-UDNS aware clients and servers

To handle the mixing of UDNS aware and non-UDNS aware clients and servers the following MUST be followed for clients and servers.

2.3.1 Native UDNS aware client

A native UDNS aware client is a client supporting all in this document.

When doing a query it MUST:
- Use the long label in the QNAME.
- If server rejected query due to long label, retry the query using the normal short label. If the QNAME contains non-ASCII it must be encoded using BCE.
- Handle answers containg BCE.

The client may skip trying a query using the long label if it knows the server does not understand it.

2.3.2 Application based UDNS aware client

An application based UDNS aware client is a client supporting UDNS through BCE handling in the application.

It only understands BCE and need only a non-UDNS aware resolver to work. All encoding and decoding of BCE is handled in the application.

Due to BCE being an ACE of BCF the names returned in an answer need not contain the real form of the name. Instead it may contains the simplified form used in name matching. As this is a transition mechanism to support non-ASCII in names before the DNS servers have been upgraded, it is acceptable and will give people a reason to upgrade.

2.3.3 non-UDNS aware client

A non-UDNS aware client will send ASCII or whatever is sent from an application. It can be BCE which will for the client just be ASCII text.

2.3.4 UDNS aware server

An UDNS aware server MUST handle all in this document and follow:

- If an incoming query contains a long label the answer may contain a long label and the client is identified as being UDNS aware.
- If the query comes from a non-UDNS aware client and the answer contains non-ASCII, the non-ASCII labels must be encoded using BCE.
- If a short label is used in a query and the QNAME contains non-ASCII, an authorative server must handle the query if the character encoding can be recognised. If must recognise SCE and should recognise common encodings used for the labels in the domain it is authorative for. Answers will use BCE for all labels except the one matching QNAME. This will allow clients using the local character set to work in many cases before the resolver code is upgraded.

2.3.5 non-UDNS aware server

A non-UDNS server can only handle ASCII matching when comparing names. It can support the transition mechanism with BCE. The authorative zones will then have to be loaded with manually BCE encoded names.

2.4 DNSSEC

As labels now can have non-ASCII in them, DNSSEC [RFC2535] need to be revised so that it also can handle that.

3. Effect on other protocols

As now a domain name may include non-ASCII many other protocols that include domain names need to be updated. For example SMTP, HTTP and URIs. The BCE format can be used when interfacing with ASCII only software or protocols. Protocols like SMTP could be extended using ESMTP and a UTF8 option that defines that all headers are in UTF-8.

It is recommended that protocols updated to handle i18n do this by encoding character data in the same standard format as defined for DNS in this document (UCS normalised form C). The use of encoding it in ASCII or by tagged character sets should be avoided.

DNS do not only have domain names in them, for example e-mail

addresses are also included. So an e-mail address would be expected to be changed to include non-ASCII both before and after the @-sign.

Software need to be updated to follow the user interface recommendations given above, so that a human will see the characters in their local character set, if possible.

4. Security Considerations

As always with data, if software does not check for data that can be a problem, security may be affected. As more characters than ASCII is allowed, software only expecting ASCII and with no checks may now get security problems.

5. References

[RFC1034] P. Mockapetris, "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987.

[RFC1035] P. Mockapetris, "Domain Names - Implementation and Specification", STD 13, RFC 1035, November 1987.

[RFC2119] Scott Bradner, "Key words for use in RFCs to Indicate Requirement Levels", March 1997, RFC 2119.

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

[RFC2279] F. Yergeau, "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

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

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

[ISO10646] ISO/IEC 10646-1:2000. International Standard -- Information technology -- Universal Multiple-Octet Coded Character Set (UCS)

[Unicode] The Unicode Consortium, "The Unicode Standard -- Version 3.0", ISBN 0-201-61633-5. Described at http://www.unicode.org/unicode/standard/versions/Unicode3.0.html

[UTR15] M. Davis and M. Duerst, "Unicode Normalization Forms", Unicode Technical Report #15, Nov 1999,

http://www.unicode.org/unicode/reports/tr15/.

[UTR21] M. Davis, "Case Mappings", Unicode Technical Report #21, Dec 1999, http://www.unicode.org/unicode/reports/tr21/.

[UDATA] The Unicode Character Database, ftp://ftp.unicode.org/Public/UNIDATA/UnicodeData.txt. The database is described in ftp://ftp.unicode.org/Public/UNIDATA/UnicodeCharacterDatabase.html.

[IDNREQ] James Seng, "Requirements of Internationalized Domain Names", draft-ietf-idn-requirement.

[IANADNS] Donald Eastlake, Eric Brunner, Bill Manning, "Domain Name System (DNS) IANA Considerations",draft-ietf-dnsext-iana-dns.

[IDNE] Marc Blanchet,Paul Hoffman, "Internationalized domain names using EDNS (IDNE)", draft-ietf-idn-idne.

[CHNORM] M. Duerst, M. Davis, "Character Normalization in IETF Protocols", draft-duerst-i18n-norm.

[IDNCOMP] Paul Hoffman, "Comparison of Internationalized Domain Name Proposals", draft-ietf-idn-compare.

[NAMEPREP] Paul Hoffman, "Comparison of Internationalized Domain Name Proposals", draft-ietf-idn-compare.

[SACE] Dan Oscarsson, "Simple ASCII Compatible Encoding", draft-ietf-idn-sace.

[RACE] Paul Hoffman, "RACE: Row-based ASCII Compatible Encoding for IDN", draft-ietf-idn-race.

6. Acknowledgements

Paul Hoffman giving many comments in our e-mail discussions.

Ideas from drafts by Paul Hoffman, Stuart Kwan, James Gilroy and Kent Karlsson.

Magnus Gustavsson, Mark Davis, Kent Karlsson and Andrew Draper for comments on my draft.

Discussions and comments by the members of the IDN working group.

Author's Address

Dan Oscarsson
Telia ProSoft AB
Box 85
201 20 Malmo
Sweden

E-mail: Dan.Oscarsson@trab.se