         The Naming Authority Pointer (NAPTR) DNS Resource Record

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

   This document describes a Domain Name System (DNS) resource record
   which specifies a regular expression based rewrite rule that, when
   applied to an existing string, will produce a new domain label or
   Uniform Resource Identifier (URI). Depending on the value of the
   flags field of the resource record, the resulting domain label or URI
   may be used in subsequent queries for the Naming Authority Pointer
   (NAPTR) resource records (to delegate the name lookup) or as the
   output of the entire process for which this system is used (a
   resolution server for URI resolution, a service URI for ENUM style
   e.164 number to URI mapping, etc).

   This allows the DNS to be used to lookup services for a wide variety
   of resource names (including URIs) which are not in domain name
   syntax. Reasons for doing this range from URN Resource Discovery
   Systems to moving out-of-date services to new domains.

   This document updates the portions of RFC 2168 specifically dealing
   with the definition of the NAPTR records and how other, non-URI
   specific applications, might use NAPTR.

Table of Contents

1. Introduction

   This RR was originally produced by the URN Working Group [3] as a way
   to encode rule-sets in DNS so that the delegated sections of a URI
   could be decomposed in such a way that they could be changed and re-
   delegated over time. The result was a Resource Record that included
   a regular expression that would be used by a client program to
   rewrite a string into a domain name. Regular expressions were chosen
   for their compactness to expressivity ratio allowing for a great deal
   of information to be encoded in a rather small DNS packet.

   The function of rewriting a string according to the rules in a record
   has usefulness in several different applications. This document
   defines the basic assumptions to which all of those applications must
   adhere to. It does not define the reasons the rewrite is used, what
   the expected outcomes are, or what they are used for. Those are
   specified by applications that define how they use the NAPTR record
   and algorithms within their contexts.

   Flags and other fields are also specified in the RR to control the
   rewrite procedure in various ways or to provide information on how to
   communicate with the host at the domain name that was the result of
   the rewrite.

The final result is a RR that has several fields that interact in a
non-trivial but implementable way. This document specifies those
fields and their values.

This document does not define applications that utilizes this rewrite
functionality. Instead it specifies just the mechanics of how it is
done. Why its done, what the rules concerning the inputs, and the
types of rules used are reserved for other documents that fully
specify a particular application. This separation is due to several
different applications all wanting to take advantage of the rewrite
rule lookup process. Each one has vastly different reasons for why
and how it uses the service, thus requiring that the definition of
the service be generic.

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.

All references to Uniform Resource Identifiers in this document
adhere to the 'absoluteURI' production of the "Collected ABNF"
found in RFC 2396 [9]. Specifically, the semantics of URI
References do not apply since the concept of a Base makes no sense
here.

2. NAPTR RR Format

The format of the NAPTR RR is given below. The DNS type code [1] for
NAPTR is 35.

Domain TTL Class Type Order Preference Flags Service Regexp
Replacement

Domain
   The domain name to which this resource record refers. This is the
   'key' for this entry in the rule database. This value will either
   be the first well known key (<something>.uri.arpa for example) or
   a new key that is the output of a replacement or regexp rewrite.
   Beyond this, it has the standard DNS requirements [1].

TTL
   Standard DNS meaning [1].

Class
   Standard DNS meaning [1].

Type
   The Type Code [1] for NAPTR is 35.

Order
    A 16-bit unsigned integer specifying the order in which the NAPTR
    records MUST be processed to ensure the correct ordering of
    rules. Low numbers are processed before high numbers, and once a
    NAPTR is found whose rule "matches" the target, the client MUST
    NOT consider any NAPTRs with a higher value for order (except as
    noted below for the Flags field).

Preference
    A 16-bit unsigned integer that specifies the order in which NAPTR
    records with equal "order" values SHOULD be processed, low
    numbers being processed before high numbers. This is similar to
    the preference field in an MX record, and is used so domain
    administrators can direct clients towards more capable hosts or
    lighter weight protocols. A client MAY look at records with
    higher preference values if it has a good reason to do so such as
    not understanding the preferred protocol or service.

    The important difference between Order and Preference is that
    once a match is found the client MUST NOT consider records with a
    different Order but they MAY process records with the same Order
    but different Preferences. I.e., Preference is used to give weight
    to rules that are considered the same from an authority
    standpoint but not from a simple load balancing standpoint.

Flags
    A <character-string> containing flags to control aspects of the
    rewriting and interpretation of the fields in the record. Flags
    are single characters from the set [A-Z0-9]. The case of the
    alphabetic characters is not significant.

    At this time only four flags, "S", "A", "U", and "P", are
    defined. The "S", "A" and "U" flags denote a terminal lookup.
    This means that this NAPTR record is the last one and that the
    flag determines what the next stage should be. The "S" flag
    means that the next lookup should be for SRV records [4]. See
    Section 5 for additional information on how NAPTR uses the SRV
    record type. "A" means that the next lookup should be for either
    an A, AAAA, or A6 record. The "U" flag means that the next step
    is not a DNS lookup but that the output of the Regexp field is an
    URI that adheres to the 'absoluteURI' production found in the
    ABNF of RFC 2396 [9]. Since there may be applications that use
    NAPTR to also lookup aspects of URIs, implementors should be
    aware that this may cause loop conditions and should act
    accordingly.

The "P" flag says that the remainder of the application side
algorithm shall be carried out in a Protocol-specific fashion.
The new set of rules is identified by the Protocol specified in
the Services field. The record that contains the 'P' flag is the
last record that is interpreted by the rules specified in this
document. The new rules are dependent on the application for
which they are being used and the protocol specified. For
example, if the application is a URI RDS and the protocol is WIRE
then the new set of rules are governed by the algorithms
surrounding the WIRE HTTP specification and not this document.

The remaining alphabetic flags are reserved for future versions
of the NAPTR specification. The numeric flags may be used for
local experimentation. The S, A, U and P flags are all mutually
exclusive, and resolution libraries MAY signal an error if more
than one is given. (Experimental code and code for assisting in
the creation of NAPTRs would be more likely to signal such an
error than a client such as a browser). It is anticipated that
multiple flags will be allowed in the future, so implementers
MUST NOT assume that the flags field can only contain 0 or 1
characters. Finally, if a client encounters a record with an
unknown flag, it MUST ignore it and move to the next record. This
test takes precedence even over the "order" field. Since flags
can control the interpretation placed on fields, a novel flag
might change the interpretation of the regexp and/or replacement
fields such that it is impossible to determine if a record
matched a given target.

The "S", "A", and "U" flags are called 'terminal' flags since
they halt the looping rewrite algorithm. If those flags are not
present, clients may assume that another NAPTR RR exists at the
domain name produced by the current rewrite rule. Since the "P"
flag specifies a new algorithm, it may or may not be 'terminal'.
Thus, the client cannot assume that another NAPTR exists since
this case is determined elsewhere.

DNS servers MAY interpret these flags and values and use that
information to include appropriate SRV and A,AAAA, or A6 records
in the additional information portion of the DNS packet. Clients
are encouraged to check for additional information but are not
required to do so.

Service
   Specifies the service(s) available down this rewrite path. It may
   also specify the particular protocol that is used to talk with a
   service. A protocol MUST be specified if the flags field states
   that the NAPTR is terminal. If a protocol is specified, but the
   flags field does not state that the NAPTR is terminal, the next

lookup MUST be for a NAPTR. The client MAY choose not to perform
the next lookup if the protocol is unknown, but that behavior
MUST NOT be relied upon.

The service field may take any of the values below (using the
Augmented BNF of RFC 2234 [5]):

```
        service_field = [ [protocol] *("+" rs)]
        protocol      = ALPHA *31ALPHANUM
        rs            = ALPHA *31ALPHANUM
        ; The protocol and rs fields are limited to 32
        ; characters and must start with an alphabetic.
```

For example, an optional protocol specification followed by 0 or
more resolution services. Each resolution service is indicated by
an initial '+' character.

Note that the empty string is also a valid service field. This
will typically be seen at the beginning of a series of rules,
when it is impossible to know what services and protocols will be
offered by a particular service.

The actual format of the service request and response will be
determined by the resolution protocol, and is the subject for
other documents. Protocols need not offer all services. The
labels for service requests shall be formed from the set of
characters [A-Z0-9]. The case of the alphabetic characters is
not significant.

The list of "valid" protocols for any given NAPTR record is any
protocol that implements some or all of the services defined for
a NAPTR application. Currently, THTTP [6] is the only protocol
that is known to make that claim at the time of publication. Any
other protocol that is to be used must have documentation
specifying:

* how it implements the services of the application

* how it is to appear in the NAPTR record (i.e., the string id
   of the protocol)

The list of valid Resolution Services is defined by the documents
that specify individual NAPTR based applications.

It is worth noting that the interpretation of this field is
subject to being changed by new flags, and that the current
specification is oriented towards telling clients how to talk
with a URN resolver.

   Regexp
      A STRING containing a substitution expression that is applied to
      the original string held by the client in order to construct the
      next domain name to lookup. The grammar of the substitution
      expression is given in the next section.

      The regular expressions MUST NOT be used in a cumulative fashion,
      that is, they should only be applied to the original string held
      by the client, never to the domain name produced by a previous
      NAPTR rewrite. The latter is tempting in some applications but
      experience has shown such use to be extremely fault sensitive,
      very error prone, and extremely difficult to debug.

   Replacement
      The next NAME to query for NAPTR, SRV, or address records
      depending on the value of the flags field. This MUST be a fully
      qualified domain-name. Unless and until permitted by future
      standards action, name compression is not to be used for this
      field.

3. Substitution Expression Grammar

   The content of the regexp field is a substitution expression. True
   sed(1) and Perl style substitution expressions are not appropriate
   for use in this application for a variety of reasons stemming from
   internationalization requirements and backref limitations, therefore
   the contents of the regexp field MUST follow the grammar below:

```
subst_expr   = delim-char  ere  delim-char  repl  delim-char  *flags
delim-char   = "/" / "!" / ... <Any non-digit or non-flag character
               other than backslash '\'. All occurances of a delim_char
               in a subst_expr must be the same character.>
ere          = POSIX Extended Regular Expression
repl         = 1 * ( OCTET /  backref )
backref      = "\" 1POS_DIGIT
flags        = "i"
POS_DIGIT    = %x31-39                  ; 0 is not an allowed backref
```

   The definition of a POSIX Extended Regular Expression can be found in
   [8], section 2.8.4.

   The result of applying the substitution expression to the original
   URI MUST result in either a string that obeys the syntax for DNS
   domain-names [1] or a URI [9] if the Flags field contains a 'u'.
   Since it is possible for the regexp field to be improperly specified,
   such that a non-conforming domain-name can be constructed, client
   software SHOULD verify that the result is a legal DNS domain-name
   before making queries on it.

Backref expressions in the repl portion of the substitution
expression are replaced by the (possibly empty) string of characters
enclosed by '(' and ')' in the ERE portion of the substitution
expression. N is a single digit from 1 through 9, inclusive. It
specifies the N'th backref expression, the one that begins with the
N'th '(' and continues to the matching ')'. For example, the ERE

                    (A(B(C)DE)(F)G)

     has backref expressions:

                    \1 = ABCDEFG
                    \2 = BCDE
                    \3 = C
                    \4 = F
                    \5..\9 = error - no matching subexpression

The "i" flag indicates that the ERE matching SHALL be performed in a
case-insensitive fashion. Furthermore, any backref replacements MAY
be normalized to lower case when the "i" flag is given.

The first character in the substitution expression shall be used as
the character that delimits the components of the substitution
expression. There must be exactly three non-escaped occurrences of
the delimiter character in a substitution expression. Since escaped
occurrences of the delimiter character will be interpreted as
occurrences of that character, digits MUST NOT be used as delimiters.
Backrefs would be confused with literal digits were this allowed.
Similarly, if flags are specified in the substitution expression, the
delimiter character must not also be a flag character.

4. The Basic NAPTR Algorithm

The behavior and meaning of the flags and services assume an
algorithm where the output of one rewrite is a new key that points to
another rule. This looping algorithm allows NAPTR records to
incrementally specify a complete rule. These incremental rules can
be delegated which allows other entities to specify rules so that one
entity does not need to understand _all_ rules.

The algorithm starts with a string and some known key (domain).
NAPTR records for this key are retrieved, those with unknown Flags or
inappropriate Services are discarded and the remaining records are
sorted by their Order field. Within each value of Order, the records
are further sorted by the Preferences field.

The records are examined in sorted order until a matching record is
found. A record is considered a match iff:

o it has a Replacement field value instead of a Regexp field value.

o or the Regexp field matches the string held by the client.

The first match MUST be the match that is used. Once a match is
found, the Services field is examined for whether or not this rule
advances toward the desired result. If so, the rule is applied to
the target string. If not, the process halts. The domain that
results from the regular expression is then used as the domain of the
next loop through the NAPTR algorithm. Note that the same target
string is used throughout the algorithm.

This looping is extremely important since it is the method by which
complex rules are broken down into manageable delegated chunks. The
flags fields simply determine at which point the looping should stop
(or other specialized behavior).

Since flags are valid at any level of the algorithm, the degenerative
case is to never loop but to look up the NAPTR and then stop. In
many specialized cases this is all that is needed. Implementors
should be aware that the degenerative case should not become the
common case.

5. Concerning How NAPTR Uses SRV Records

When the SRV record type was originally specified it assumed that the
client did not know the specific domain-name before hand. The client
would construct a domain-name more in the form of a question than the
usual case of knowing ahead of time that the domain-name should
exist. I.e., if the client wants to know if there is a TCP based
HTTP server running at a particular domain, the client would
construct the domain-name _http._tcp.somedomain.com and ask the DNS
if that records exists. The underscores are used to avoid collisions
with potentially 'real' domain-names.

In the case of NAPTR, the actual domain-name is specified by the
various fields in the NAPTR record. In this case the client isn't
asking a question but is instead attempting to get at information
that it has been told exists in an SRV record at that particular
domain-name. While this usage of SRV is slightly different than the
SRV authors originally intended it does not break any of the
assumptions concerning what SRV contains. Also, since the NAPTR
explicitly spells out the domain-name for which an SRV exists, that
domain-name MUST be used in SRV queries with NO transformations. Any
given NAPTR record may result in a domain-name to be used for SRV
queries that may or may not contain the SRV standardized underscore

characters. NAPTR applications that make use of SRV MUST NOT attempt
to understand these domains or use them according to how the SRV
specification structures its query domains.

6. Application Specifications

It should be noted that the NAPTR algorithm is the basic assumption
about how NAPTR works. The reasons for the rewrite and the expected
output and its use are specified by documents that define what
applications the NAPTR record and algorithm are used for. Any
document that defines such an application must define the following:

o The first known domain-name or how to build it

o The valid Services and Protocols

o What the expected use is for the output of the last rewrite

o The validity and/or behavior of any 'P' flag protocols.

o The general semantics surrounding why and how NAPTR and its
   algorithm are being used.

7. Examples

NOTE: These are examples only. They are taken from ongoing work and
may not represent the end result of that work. They are here for
pedagogical reasons only.

7.1 Example 1

NAPTR was originally specified for use with the a Uniform Resource
Name Resolver Discovery System. This example details how a
particular URN would use the NAPTR record to find a resolver service.

Consider a URN namespace based on MIME Content-Ids. The URN might
look like this:

    urn:cid:39CB83F7.A8450130@fake.gatech.edu

(Note that this example is chosen for pedagogical purposes, and does
not conform to the CID URL scheme.)

The first step in the resolution process is to find out about the CID
namespace. The namespace identifier [3], 'cid', is extracted from
the URN, prepended to urn.arpa. 'cid.urn.arpa' then becomes the first
'known' key in the NAPTR algorithm. The NAPTR records for
cid.urn.arpa looked up and return a single record:

```
       cid.urn.arpa.
       ;;        order pref flags service        regexp          replacement
       IN NAPTR 100   10    ""    ""   "/urn:cid:.+@([^\.]+\.)(.*)$/\2/i"     .
```

   There is only one NAPTR response, so ordering the responses is not a
   problem. The replacement field is empty, so the pattern provided in
   the regexp field is used. We apply that regexp to the entire URN to
   see if it matches, which it does. The \2 part of the substitution
   expression returns the string "gatech.edu". Since the flags field
   does not contain "s" or "a", the lookup is not terminal and our next
   probe to DNS is for more NAPTR records where the new domain is '
   gatech.edu' and the string is the same string as before.

   Note that the rule does not extract the full domain name from the
   CID, instead it assumes the CID comes from a host and extracts its
   domain. While all hosts, such as mordred, could have their very own
   NAPTR, maintaining those records for all the machines at a site as
   large as Georgia Tech would be an intolerable burden. Wildcards are
   not appropriate here since they only return results when there is no
   exactly matching names already in the system.

   The record returned from the query on "gatech.edu" might look like:

```
;;         order pref flags service            regexp  replacement
 IN NAPTR 100  50   "s"  "z3950+I2L+I2C"        ""   _z3950._tcp.gatech.edu.
 IN NAPTR 100  50   "s"  "rcds+I2C"             ""   _rcds._udp.gatech.edu.
 IN NAPTR 100  50   "s"  "http+I2L+I2C+I2R"     ""   _http._tcp.gatech.edu.
```

   Continuing with the example, note that the values of the order and
   preference fields are equal in all records, so the client is free to
   pick any record. The flags field tells us that these are the last
   NAPTR patterns we should see, and after the rewrite (a simple
   replacement in this case) we should look up SRV records to get
   information on the hosts that can provide the necessary service.

   Assuming we prefer the Z39.50 protocol, our lookup might return:

```
;;                        Pref Weight   Port Target
 _z3950._tcp.gatech.edu. IN SRV 0    0      1000 z3950.gatech.edu.
                         IN SRV 0    0      1000 z3950.cc.gatech.edu.
                         IN SRV 0    0      1000 z3950.uga.edu.
```

   telling us three hosts that could actually do the resolution, and
   giving us the port we should use to talk to their Z39.50 server.

   Recall that the regular expression used \2 to extract a domain name
   from the CID, and \. for matching the literal '.' characters
   separating the domain name components. Since '\' is the escape

character, literal occurances of a backslash must be escaped by
another backslash. For the case of the cid.urn.arpa record above,
the regular expression entered into the master file should be
"/urn:cid:.+@([^\\.]+\\.)(.*)$/\\2/i". When the client code actually
receives the record, the pattern will have been converted to
"/urn:cid:.+@([^\.]+\.)(.*)$/\2/i".

7.2 Example 2

Even if URN systems were in place now, there would still be a
tremendous number of URLs. It should be possible to develop a URN
resolution system that can also provide location independence for
those URLs. This is related to the requirement that URNs be able to
grandfather in names from other naming systems, such as ISO Formal
Public Identifiers, Library of Congress Call Numbers, ISBNs, ISSNs,
etc.

The NAPTR RR could also be used for URLs that have already been
assigned. Assume we have the URL for a very popular piece of
software that the publisher wishes to mirror at multiple sites around
the world:

Using the rules specified for this application we extract the prefix,
"http", and lookup NAPTR records for http.uri.arpa. This might
return a record of the form

```
   http.uri.arpa. IN NAPTR
   ;; order   pref flags service     regexp              replacement
        100     90   ""      ""    "!http://([^/:]+)!\1!i"        .
```

This expression returns everything after the first double slash and
before the next slash or colon. (We use the '!' character to delimit
the parts of the substitution expression. Otherwise we would have to
use backslashes to escape the forward slashes and would have a regexp
in the zone file that looked like "/http:\\/\\/([^\\/:]+)/\\1/i".).

Applying this pattern to the URL extracts "www.foo.com". Looking up
NAPTR records for that might return:

```
   www.foo.com.
   ;;        order pref flags   service  regexp      replacement
    IN NAPTR 100  100  "s"   "http+I2R"   ""    _http._tcp.foo.com.
    IN NAPTR 100  100  "s"   "ftp+I2R"    ""    _ftp._tcp.foo.com.
```

Looking up SRV records for http.tcp.foo.com would return information
on the hosts that foo.com has designated to be its mirror sites. The
client can then pick one for the user.

7.3 Example 3

   A non-URI example is the ENUM application which uses a NAPTR record
   to map an e.164 telephone number to a URI. In order to convert the
   phone number to a domain name for the first iteration all characters
   other than digits are removed from the the telephone number, the
   entire number is inverted, periods are put between each digit and the
   string ".e164.arpa" is put on the left-hand side. For example, the
   E.164 phone number "+1-770-555-1212" converted to a domain-name it
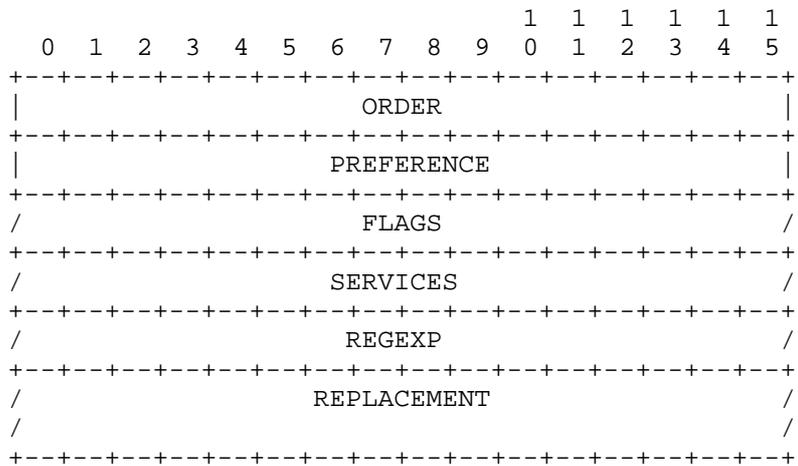   would be "2.1.2.1.5.5.5.0.7.7.1.e164.arpa."

   For this example telephone number we might get back the following
   NAPTR records:

```
$ORIGIN 2.1.2.1.5.5.5.0.7.7.1.e164.arpa.
 IN NAPTR 100 10 "u" "sip+E2U"   "!^.*$!sip:information@tele2.se!"     .
 IN NAPTR 102 10 "u" "mailto+E2U" "!^.*$!mailto:information@tele2.se!"  .
```

   This application uses the same ‘u’ flag as the URI Resolution
   application. This flag states that the Rule is terminal and that the
   output is a URI which contains the information needed to contact that
   telephone service. ENUM also uses the same format for its Service
   field except that it defines the ‘E2U’ service instead of the ‘I2*’
   services that URI resolution uses. The example above states that the
   available protocols used to access that telephone’s service are
   either the Session Initiation Protocol or SMTP mail.

8. DNS Packet Format

        The packet format for the NAPTR record is:

```
                                  1  1  1  1  1  1
          0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
        +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
        |                     ORDER                     |
        +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
        |                   PREFERENCE                  |
        +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
        /                     FLAGS                     /
        +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
        /                    SERVICES                   /
        +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
        /                     REGEXP                    /
        +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
        /                   REPLACEMENT                 /
        /                                               /
        +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

where:

FLAGS A <character-string> which contains various flags.

SERVICES A <character-string> which contains protocol and service
   identifiers.

REGEXP A <character-string> which contains a regular expression.

REPLACEMENT A <domain-name> which specifies the new value in the
   case where the regular expression is a simple replacement
   operation.

<character-string> and <domain-name> as used here are defined in
RFC1035 [1].

9. Master File Format

The master file format follows the standard rules in RFC-1035 [1].
Order and preference, being 16-bit unsigned integers, shall be an
integer between 0 and 65535. The Flags and Services and Regexp
fields are all quoted <character-string>s. Since the Regexp field
can contain numerous backslashes and thus should be treated with
care. See Section 10 for how to correctly enter and escape the
regular expression.

10. Advice for DNS Administrators

Beware of regular expressions. Not only are they difficult to get
correct on their own, but there is the previously mentioned
interaction with DNS. Any backslashes in a regexp must be entered
twice in a zone file in order to appear once in a query response.
More seriously, the need for double backslashes has probably not been
tested by all implementors of DNS servers.

The "a" flag allows the next lookup to be for address records (A,
AAAA, A6) rather than SRV records. Since there is no place for a
port specification in the NAPTR record, when the "A" flag is used the
specified protocol must be running on its default port.

The URN Syntax draft defines a canonical form for each URN, which
requires %encoding characters outside a limited repertoire. The
regular expressions MUST be written to operate on that canonical
form. Since international character sets will end up with extensive
use of %encoded characters, regular expressions operating on them
will be essentially impossible to read or write by hand.

11. Notes

   o A client MUST process multiple NAPTR records in the order
      specified by the "order" field, it MUST NOT simply use the first
      record that provides a known protocol and service combination.

   o When multiple RRs have the same "order" and all other criteria
      being equal, the client should use the value of the preference
      field to select the next NAPTR to consider. However, because it
      will often be the case where preferred protocols or services
      exist, clients may use this additional criteria to sort
      the records.

   o If the lookup after a rewrite fails, clients are strongly
      encouraged to report a failure, rather than backing up to pursue
      other rewrite paths.

   o Note that SRV RRs impose additional requirements on clients.

12. IANA Considerations

   The only registration function that impacts the IANA is for the
   values that are standardized for the Services and Flags fields. To
   extend the valid values of the Flags field beyond what is specified
   in this document requires a published specification that is approved
   by the IESG.

   The values for the Services field will be determined by the
   application that makes use of the NAPTR record. Those values must be
   specified in a published specification and approved by the IESG.

13. Security Considerations

   The interactions with DNSSEC are currently being studied. It is
   expected that NAPTR records will be signed with SIG records once the
   DNSSEC work is deployed.

   The rewrite rules make identifiers from other namespaces subject to
   the same attacks as normal domain names. Since they have not been
   easily resolvable before, this may or may not be considered a
   problem.

   Regular expressions should be checked for sanity, not blindly passed
   to something like PERL.

   This document has discussed a way of locating a service, but has not
   discussed any detail of how the communication with that service takes
   place. There are significant security considerations attached to the

communication with a service. Those considerations are outside the
scope of this document, and must be addressed by the specifications
for particular communication protocols.

14. Acknowledgments

The editors would like to thank Keith Moore for all his consultations
during the development of this memo. We would also like to thank
Paul Vixie for his assistance in debugging our implementation, and
his answers on our questions. Finally, we would like to acknowledge
our enormous intellectual debt to the participants in the Knoxville
series of meetings, as well as to the participants in the URI and URN
working groups.

References

    [1] Mockapetris, P., "Domain names - implementation and
         specification", STD 13, RFC 1035, November 1987.

    [2] Mockapetris, P., "Domain names - concepts and facilities", STD
         13, RFC 1034, November 1987.

    [3] Moats, R., "URN Syntax", RFC 2141, May 1997.

    [4] Gulbrandsen, A., Vixie, P. and L. Esibov, "A DNS RR for
         specifying the location of services (DNS SRV)", RFC 2782,
         February 2000.

    [5] Crocker, D., "Augmented BNF for Syntax Specifications: ABNF",
         RFC 2234, November 1997.

    [6] Daniel, R., "A Trivial Convention for using HTTP in URN
         Resolution", RFC 2169, June 1997.

    [7] Daniel, R. and M. Mealling, "Resolution of Uniform Resource
         Identifiers using the Domain Name System", RFC 2168, June 1997.

    [8] IEEE, "IEEE Standard for Information Technology - Portable
         Operating System Interface (POSIX) - Part 2: Shell and Utilities
         (Vol. 1)", IEEE Std 1003.2-1992, January 1993.

    [9] Berners-Lee, T., Fielding, R.T. and L. Masinter, "Uniform
         Resource Identifiers (URI): Generic Syntax", RFC 2396, August
         1998.

Authors' Addresses

   Michael Mealling
   Network Solutions, Inc.
   505 Huntmar Park Drive
   Herndon, VA 22070
   US

   Phone: +1 770 921 2251
   EMail: michaelm@netsol.com
   URI: http://www.netsol.com

   Ron Daniel
   DATAFUSION, Inc.
   139 Townsend Street, Ste. 100
   San Francisco, CA 94107
   US

   Phone: +1 415 222 0100
   EMail: rdaniel@datafusion.net
   URI: http://www.datafusion.net

Full Copyright Statement

Acknowledgement