

Network Working Group
Request for Comments: 3403
Obsoletes: 2915, 2168
Category: Standards Track

M. Mealling
VeriSign
October 2002

Dynamic Delegation Discovery System (DDDS)
Part Three: The Domain Name System (DNS) Database

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

Copyright (C) The Internet Society (2002). All Rights Reserved.

Abstract

This document describes a Dynamic Delegation Discovery System (DDDS) Database using the Domain Name System (DNS) as a distributed database of Rules. The Keys are domain-names and the Rules are encoded using the Naming Authority Pointer (NAPTR) Resource Record (RR).

Since this document obsoletes RFC 2915, it is the official specification for the NAPTR DNS Resource Record. It is also part of a series that is completely specified in "Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS" (RFC 3401). It is very important to note that it is impossible to read and understand any document in this series without reading the others.

Table of Contents

1.	Introduction	2
2.	Terminology	3
3.	DDDS Database Specification	3
4.	NAPTR RR Format	5
4.1	Packet Format	5
4.2	Additional Information Processing	7
4.2.1	Additional Section processing by DNS servers	7
4.2.2	Additional Section processing by resolver/applications	7
4.3	Master File Format	7
5.	Application Specifications	8
6.	Examples	8
6.1	URN Example	8
6.2	E164 Example	10
7.	Advice for DNS Administrators	10
8.	Notes	11
9.	IANA Considerations	11
10.	Security Considerations	11
	References	12
	Author's Address	13
	Full Copyright Statement	14

1. Introduction

The Dynamic Delegation Discovery System (DDDS) is used to implement lazy binding of strings to data, in order to support dynamically configured delegation systems. The DDDS functions by mapping some unique string to data stored within a DDDS Database by iteratively applying string transformation rules until a terminal condition is reached.

This document describes the way in which the Domain Name System (DNS) is used as a data store for the Rules that allow a DDDS Application to function. It does not specify any particular application or usage scenario. The entire series of documents is specified in "Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS" (RFC 3401) [1]. It is very important to note that it is impossible to read and understand any document in that series without reading the related documents.

The Naming Authority Pointer (NAPTR) DNS Resource Record (RR) specified here was originally produced by the URN Working Group as a way to encode rule-sets in DNS so that the delegated sections of a Uniform Resource Identifiers (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.

Over time this process was generalized for other Applications and Rule Databases. This document defines a Rules Database absent any particular Application as there may be several Applications all taking advantage of this particular Rules Database.

2. 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 [6].

All other terminology, especially capitalized terms, is taken from [3].

3. DDDS Database Specification

General Description:

This database uses the Domain Name System (DNS) as specified in [8] and [7].

The character set used to specify the various values of the NAPTR records is UTF-8 [17]. Care must be taken to ensure that, in the case where either the input or the output to the substitution expression contains code points outside of the ASCII/Unicode equivalence in UTF-8, any UTF-8 is interpreted as a series of code-points instead of as a series of bytes. This is to ensure that the internationalized features of the POSIX Extended Regular Expressions are able to match their intended code-points. Substitution expressions MUST NOT be written where they depend on a specific POSIX locale since this would cause substitution expressions to lose their ability to be universally applicable.

All DNS resource records have a Time To Live (TTL) associated with them. When the number of seconds has passed since the record was retrieved the record is no longer valid and a new query must be used to retrieve the new records. Thus, as mentioned in the DDDS Algorithm, there can be the case where a given Rule expires. In the case where an application attempts to fall back to previously retrieved sets of Rules (either in the case of a bad delegation path or some network or server failure) the application MUST ensure that none of the records it is relying on have expired. In the case where even a single record has expired, the application is required to start over at the beginning of the algorithm.

Key Format:

A Key is a validly constructed DNS domain-name.

Lookup Request:

In order to request a set of rules for a given Key, the client issues a request, following standard DNS rules, for NAPTR Resource Records for the given domain-name.

Lookup Response:

The response to a request for a given Key (domain-name) will be a series of NAPTR records. The format of a NAPTR Resource Record can be found in Section 4.

Rule Insertion Procedure:

Rules are inserted by adding new records to the appropriate DNS zone. If a Rule produces a Key that exists in a particular zone then only the entity that has administrative control of that zone can specify the Rule associated with that Key.

Collision Avoidance:

In the case where two Applications may use this Database (which is actually the case with the ENUM and URI Resolution Applications, Section 6.2), there is a chance of collision between rules where two NAPTR records appear in the same domain but they apply to more than one Application. There are three ways to avoid collisions:

- * create a new zone within the domain in common that contains only NAPTR records that are appropriate for the application. E.g., all URI Resolution records would exist under `urires.example.com` and all ENUM records would be under `enum.example.com`. In the case where this is not possible due to lack of control over the upstream delegation the second method is used.
- * write the regular expression such that it contains enough of the Application Unique string to disambiguate it from any other. For example, the URI Resolution Application would be able to use the scheme name on the left hand side to anchor the regular expression match to that scheme. An ENUM specific record in that same zone would be able to anchor the left hand side of the match with the "+" character which is defined by ENUM to be at the beginning of every Application Unique String. This way a given Application Unique String can only match one or the other record, not both.
- * if two Application use different Flags or Services values then a record from another Application will be ignored since it doesn't apply to the Services/Flags in question.

4. NAPTR RR Format

4.1 Packet Format

The packet format of the NAPTR RR is given below. The DNS type code for NAPTR is 35.

The packet format for the NAPTR record is as follows

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

<character-string> and <domain-name> as used here are defined in RFC 1035 [7].

ORDER

A 16-bit unsigned integer specifying the order in which the NAPTR records MUST be processed in order to accurately represent the ordered list of Rules. The ordering is from lowest to highest. If two records have the same order value then they are considered to be the same rule and should be selected based on the combination of the Preference values and Services offered.

PREFERENCE

Although it is called "preference" in deference to DNS terminology, this field is equivalent to the Priority value in the DDDS Algorithm. It is 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 supporting some protocol or service very well.

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. The only exception to this is noted in the second important Note in the DDDS algorithm specification concerning allowing clients to use more complex Service determination between steps 3 and 4 in the algorithm. Preference is used to give communicate a higher quality of service to rules that are considered the same from an authority standpoint but not from a simple load balancing standpoint.

It is important to note that DNS contains several load balancing mechanisms and if load balancing among otherwise equal services should be needed then methods such as SRV records or multiple A records should be utilized to accomplish load balancing.

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-Z and 0-9. The case of the alphabetic characters is not significant. The field can be empty.

It is up to the Application specifying how it is using this Database to define the Flags in this field. It must define which ones are terminal and which ones are not.

SERVICES

A <character-string> that specifies the Service Parameters applicable to this this delegation path. It is up to the Application Specification to specify the values found in this field.

REGEXP

A <character-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. See the DDDS Algorithm specification for the syntax of this field.

As stated in the DDDS algorithm, 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

A <domain-name> which is the next domain-name to query for depending on the potential values found in the flags field. This field is used when the regular expression is a simple replacement operation. Any value in this field **MUST** be a fully qualified domain-name. Name compression is not to be used for this field.

This field and the REGEXP field together make up the Substitution Expression in the DDDS Algorithm. It is simply a historical optimization specifically for DNS compression that this field exists. The fields are also mutually exclusive. If a record is returned that has values for both fields then it is considered to be in error and **SHOULD** be either ignored or an error returned.

4.2 Additional Information Processing

Additional section processing requires upgraded DNS servers, thus it will take many years before applications can expect to see relevant records in the additional information section.

4.2.1 Additional Section Processing by DNS Servers

DNS servers **MAY** add RRsets to the additional information section that are relevant to the answer and have the same authenticity as the data in the answer section. Generally this will be made up of A and SRV records but the exact records depends on the application.

4.2.2 Additional Section Processing by Resolver/Applications

Applications **MAY** inspect the Additional Information section for relevant records but Applications **MUST NOT** require that records of any type be in the Additional Information section of any DNS response in order for clients to function. All Applications must be capable of handling responses from nameservers that never fill in the Additional Information part of a response.

4.3 Master File Format

The master file format follows the standard rules in RFC-1035. 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 7 for how to correctly enter and escape the regular expression.

5. Application Specifications

This DDDS Database is usable by any application that makes use of the DDDS algorithm. In addition to the items required to specify a DDDS Application, an application wishing to use this Database must also define the following values:

- o What domain the Key that is produced by the First Well Known Rule belongs to. Any application must ensure that its rules do not collide with rules used by another application making use of this Database. For example, the 'foo' application might have all of its First Well Known Keys be found in the 'foo.net' zone.
- o What the allowed values for the Services and Protocols fields are.
- o What the expected output is of the terminal rewrite rule in addition to how the Flags are actually encoded and utilized.

6. Examples

6.1 URN Example

The NAPTR record was originally created for use with the Uniform Resource Name (URN) Resolver Discovery Service (RDS) [15]. This example details how a particular URN would use the NAPTR record to find a resolver service that can answer questions about the URN. See [2] for the definitive specification for this Application.

Consider a URN namespace based on MIME Content-Ids (this is very hypothetical so do not rely on this). The URN might look like this:

```
urn:cid:199606121851.1@bar.example.com
```

This Application's First Well Known Rule is to extract the characters between the first and second colon. For this URN that would be 'cid'. The Application also specifies that, in order to build a Database-valid Key, the string 'urn.arpa' should be appended to the result of the First Well Known Rule. The result is 'cid.urn.arpa'. Next, the client queries the DNS for NAPTR records for the domain-name 'cid.urn.arpa'. The result is a single record:

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

Since there is only one record, 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 "example.com". Since the flags field is empty, the lookup is not terminal and our next probe to DNS is for more NAPTR records where the new domain is 'example.com'.

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 'bar', could have their very own NAPTR, maintaining those records for all the machines at a site could 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 "example.com" might look like:

```
example.com.
;;      order pref flags service      regexp replacement
IN NAPTR 100 50 "a"  "z3950+N2L+N2C"  ""  cidserver.example.com.
IN NAPTR 100 50 "a"  "rcds+N2C"      ""  cidserver.example.com.
IN NAPTR 100 50 "s"  "http+N2L+N2C+N2R" ""  www.example.com.
```

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 Application defines the flag 'a' to mean a terminal lookup and that the output of the rewrite will be a domain-name for which an A record should be queried. Once the client has done that, it has the following information: the host, its IP address, the protocol, and the services available via that protocol. Given these bits of information the client has enough to be able to contact that server and ask it questions about the URN.

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 occurrences 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".

6.2 E164 Example

The ENUM Working Group in the IETF has specified a service that allows a telephone number to be mapped to a URI [18]. The Application Unique String for the ENUM Application is the E.164 telephone number with the dashes removed. The First Well Known Rule is to remove all characters from the telephone number and then use the entire number as the first Key. For example, the phone number "770-555-1212" represented as an E.164 number would be "+1-770-555-1212". Converted to the Key it would be "17705551212".

The ENUM Application at present only uses this Database. It specifies that, in order to convert the first Key into a form valid for this Database, periods are inserted between each digit, the entire Key is inverted and then "e164.arpa" is appended to the end. The above telephone number would then read "2.1.2.1.5.5.5.0.7.7.1.e164.arpa.". This domain-name is then used to retrieve Rewrite Rules as NAPTR records.

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@foo.se!i" .
IN NAPTR 102 10 "u" "smtp+E2U" "!.^.*$!mailto:information@foo.se!i" .
```

Both the ENUM [18] and URI Resolution [4] Applications use the 'u' flag. 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 Parameters. These state that the available protocols used to access that telephone's service are either the Session Initiation Protocol or SMTP mail.

7. 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.

In order to mitigate zone file problems, administrators should encourage those writing rewrite rules to utilize the 'default delimiter' feature of the regular expression. In the DDDS specification the regular expression starts with the character that is to be the delimiter. Hence if the first character of the regular expression is an exclamation mark ('!') for example then the regular expression can usually be written with fewer backslashes.

8. Notes

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 Service Parameter combination.

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.

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

9. IANA Considerations

The values for the Services and Flags fields will be determined by the Application that makes use of this DDDS Database. Those values may require a registration mechanism and thus may need some IANA resources. This specification by itself does not.

10. Security Considerations

The NAPTR record, like any other DNS record, can be signed and validated according to the procedures specified in DNSSEC.

This Database makes 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 since arbitrary code can be included and subsequently processed.

References

- [1] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS", RFC 3401, October 2002.
- [2] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Two: The Algorithm", RFC 3402, October 2002.
- [3] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database", RFC 3403, October 2002.
- [4] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Four: The Uniform Resource Identifiers (URI) Resolution Application", RFC 3404, October 2002.
- [5] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Five: URI.ARPA Assignment Procedures", RFC 3405, October 2002.
- [6] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [7] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [8] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [9] Gulbrandsen, A., Vixie, P. and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.
- [10] Crocker, D., "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.
- [11] Daniel, R., "A Trivial Convention for using HTTP in URN Resolution", RFC 2169, June 1997.
- [12] 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.
- [13] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.
- [14] Moats, R., "URN Syntax", RFC 2141, May 1997.

- [15] Sollins, K., "Architectural Principles of Uniform Resource Name Resolution", RFC 2276, January 1998.
- [16] Daniel, R. and M. Mealling, "Resolution of Uniform Resource Identifiers using the Domain Name System", RFC 2168, June 1997.
- [17] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.
- [18] Faltstrom, P., "E.164 number and DNS", RFC 2916, September 2000.

Author's Address

Michael Mealling
VeriSign
21345 Ridgetop Circle
Sterling, VA 20166
US

EMail: michael@neonym.net
URI: <http://www.verisignlabs.com>

Full Copyright Statement

Copyright (C) The Internet Society (2002). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

