HTTP API for Updating DNS Records
draft-jennings-app-dns-update-02

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79. This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

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

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

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

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

This Internet-Draft will expire on September 9, 2009.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.
This specification defines a simple HTTP based scheme for clients to update DNS records.

The draft is being discussed on the apps-discuss@ietf.org list.

Legal

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (http://trustee.ietf.org/license-info). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

This specification defines a simple HTTP based scheme for clients to update DNS records.

The draft is being discussed on the apps-discuss@ietf.org list.

Legal

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (http://trustee.ietf.org/license-info). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

1. Introduction

There are many circumstances in which an application or device would like to have an easy way to update DNS records. While a number of support DNS based protocols exist for updating records [RFC2136] [RFC3007] [RFC4310], many of these mechanisms are not available in today’s scaled down applications and devices. However, many existing application and devices do support the use of HTTP [RFC2616] and HTTP over TLS to update DNS records. The goal of this specification is to create a generic standard for which applications and devices can update DNS records using HTTP over TLS.

The need for this protocol exists from the use of DHCP and other dynamic IP addressing systems, where a device receives updates to it IP address, and further, there exists a need for the global DNS to be made aware of such a change. Many residential NAT devices support this type of operation today, but do it using hap-hazard and proprietary methods [DynDNS] [telnic] [SliceHost].

The approach described in the specification allows a client to make HTTP over TLS requests to a server to update DNS records, using standard and highly available encryption techniques for security, while providing a generic a flexible interface for updating DNS.
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 RFC 2119 [RFC2119].

3. Update Record Requests

This section describes the semantics of requests to update DNS records. The specification only covers how tell a DNS system what updates are desired. How the DNS system deal with SOA records or DNSSEC if not effected in any way by this specification.

3.1. HTTPS Request

The client needs to be configured with the base URL for the server, along with a username and password. The request is created by forming an HTTPS[RFC2818] POST request to a URL. The HTTPS POST request is formed by starting with the configured base URL, and then appending all the required parameters. The request MUST be done using HTTPS to protect the password. The client MUST ensure the TLS certificate of the server is appropriately signed.

The HTTP request SHOULD contain a "User-Agent" header that clearly identifies the version of the software making the request, as this facilitates debugging.

3.2. URL Parameters

The request MUST include exactly one user, password, domain, and type parameter as defined below. Other parameters are optional and can occur at most once. The values of parameters MUST be appropriately escaped as required to be part of a valid HTTP URL.

Open Issue: there is some discussion going on around if it is better to use HTTP basic auth or form style parameters. TODO Resolve this.
### General Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>The configured user name for the user making the request.</td>
</tr>
<tr>
<td>password</td>
<td>The configured password for the user making the request base16 encoded as defined in [RFC4648].</td>
</tr>
<tr>
<td>domain</td>
<td>The fully qualified domain name for the record to update.</td>
</tr>
<tr>
<td>type</td>
<td>The ASCII encoded version of their type of DNS record to update.</td>
</tr>
<tr>
<td>rdata</td>
<td>The value that should be stored in the DNS resource record.</td>
</tr>
<tr>
<td>match</td>
<td>The value that matches an existing resource record that is to be updated by this request. A special value of &quot;*&quot; means that all existing records are replaced by the new record in this request.</td>
</tr>
<tr>
<td>ttl</td>
<td>Requested time to live for the DNS records in seconds. If omitted, this will be set to default chosen by the server.</td>
</tr>
</tbody>
</table>

Some common values for the type parameter field are shown in the following table.

### Type Parameters Values

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>DNS A record [RFC1035].</td>
</tr>
<tr>
<td>AAAA</td>
<td>DNS AAAA record [RFC3596].</td>
</tr>
<tr>
<td>CNAME</td>
<td>DNS CNAME record [RFC1035].</td>
</tr>
<tr>
<td>NS</td>
<td>DNS NS record [RFC1035].</td>
</tr>
<tr>
<td>PTR</td>
<td>DNS PTR record [RFC1035].</td>
</tr>
<tr>
<td>SRV</td>
<td>DNS SRV record [RFC2782].</td>
</tr>
<tr>
<td>TXT</td>
<td>DNS TXT record [RFC1464].</td>
</tr>
<tr>
<td>HIP</td>
<td>DNS HIP record [RFC5205].</td>
</tr>
<tr>
<td>MX</td>
<td>DNS MX record.</td>
</tr>
<tr>
<td>SPF</td>
<td>DNS SPF record.</td>
</tr>
</tbody>
</table>
3.2.1. Entries with multiple records

For many updates, where only one resource record is desired, the match parameter is sent with a value of "*" indicating all existing records are removed and replaced with the new one. Sometimes it is desirable to have multiple records of the same type for the same name. For example, a domain may have multiple MX records. To add a new record, no match value is sent, or the match value is empty, and a new record is appended to the set. To update an existing record, the match parameter is set to the value of the old record that needs to be updated. If the record in the match parameter can not be found, the request returns an 404 error.

3.2.2. Deleting records

If the value of the parameter that would update a record is empty, the record MUST be removed from DNS.

3.3. Return Codes and Errors

HTTP response codes are used to indicate success and errors as specified in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Error Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>No error, operation successful</td>
</tr>
<tr>
<td>400</td>
<td>The update parameters passed are invalid or would otherwise result in an ambiguous update</td>
</tr>
<tr>
<td>401</td>
<td>Bad authentication credentials</td>
</tr>
<tr>
<td>403</td>
<td>Trying to update a record for which the given credentials are not authorized.</td>
</tr>
<tr>
<td>404</td>
<td>No records were found that match the value in the match parameter of the request.</td>
</tr>
<tr>
<td>406</td>
<td>A valid update was passed, however, it was not accepted for reasons of update abuse, whereby excessive numbers of duplicate updates have been sent.</td>
</tr>
<tr>
<td>409</td>
<td>A valid update was passed, however, no change was made as the requested change was preexisting</td>
</tr>
<tr>
<td>501</td>
<td>The server does not support the specified operation</td>
</tr>
<tr>
<td>503</td>
<td>The server is too busy to service the request or is otherwise unavailable and the client should wait at least 5 minutes before trying to update again</td>
</tr>
</tbody>
</table>

The body of the response MAY have human readable text that allows a
network administrator to learn more about why the request failed.

4. Example

In the examples below, some of the URLs appear broken across multiple lines. This is because of physical width limitations in this document; such URLs need to be read as single URLs with no embedded white space. All of the examples assume that a user called "me@example.net" with password "no" is allowed to update records in the example.com domain. The base URL for the DNS update service of https://dns.example.org/dns/update is used in the examples.

Each example shows the state of the DNS in a precondition before the request, the requests performed using this specification, and then the resulting state of the DNS in the postcondition.

4.1. Update an A record

This example shows a basic update where all existing A record values are replaced with a new entry.

Precondition:

www.example.com   A    192.0.2.0
www.example.com   A    192.0.2.1

Requests:

https://dns.example.org/dns/update?user=me%40example.net
  &password=6E6F&match=*&domain=www.example.com&type=A
  &rdata=192.0.2.2

Postcondition:

www.example.com   A    192.0.2.2

4.2. Create two MX records

This example shows how to create entries where there are multiple records.
Precondition:

Requests:

https://dns.example.org/dns/update?user=me@example.net
   &password=6E6F&domain=example.com&match=&type=MX
   &rdata=10%20mail1.example.com

https://dns.example.org/dns/update?user=me@example.net
   &password=6E6F&domain=example.com&match=&type=MX
   &rdata=20%20mail2.example.com

Postcondition:

  example.com  MX  10  mail1.example.com
  example.com  MX  20  mail2.example.com

4.3. Delete an A record

This example shows a simple removal of a record.

Precondition:

  www.example.com  A  192.0.2.1
  www.example.com  A  192.0.2.2
  www.example.com  A  192.0.2.3

Requests:

https://dns.example.org/dns/update?user=me@example.net
   &password=6E6F&domain=www.example.com&type=A&match=192.0.2.2

Postcondition:

  www.example.com  A  192.0.2.1
  www.example.com  A  192.0.2.3

4.4. Add a new SRV record

This example shows how to append a record to a list of existing records.

Precondition:

  _sip._tcp.example.com  SRV 10 1 5060 sip1.example.com
  _sip._tcp.example.com  SRV 10 1 5060 sip2.example.com

Requests:

https://dns.example.org/dns/update?user=me@example.net
   &password=6E6F&domain=_sip._tcp.example.com&match=&type=SRV&rdata=10%201%205060%20sip3.example.com

Postcondition:

  _sip._tcp.example.com  SRV 10 1 5060 sip1.example.com
  _sip._tcp.example.com  SRV 10 1 5060 sip2.example.com
4.5. Update an existing SRV record

This example is similar to the previous one, in that an entry is being changed.

Precondition:

_\texttt{sip.\_tcp.example.com} \texttt{ SRV 10 1 5060 sip1.example.com}
_\texttt{sip.\_tcp.example.com} \texttt{ SRV 10 1 5060 sip2.example.com}
_\texttt{sip.\_tcp.example.com} \texttt{ SRV 10 1 5060 sip3.example.com}

Requests:

\url{https://dns.example.org/dns/update?user=me@example.net}
&\texttt{password=6E6F&domain=_sip.\_tcp.example.com&type=SRV}
&\texttt{match=10\%201\%205060\%20sip2.example.com}
&\texttt{rdata=10\%201\%205060\%20sip22.example.com}

Postcondition:

_\texttt{sip.\_tcp.example.com} \texttt{ SRV 10 1 5060 sip1.example.com}
_\texttt{sip.\_tcp.example.com} \texttt{ SRV 10 1 5060 sip22.example.com}
_\texttt{sip.\_tcp.example.com} \texttt{ SRV 10 1 5060 sip3.example.com}

5. WADL Description

This section is non normative. The WADL [sun.wadl] description for the examples is:
<?xml version="1.0" encoding="utf-8"?>
<application
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xsi:schemaLocation="http://research.sun.com/wadl/2006/10 wadl.xsd"
    xmlns="http://research.sun.com/wadl/2006/10">
  <resources base="https://dns.example.org/">
    <resource path="dns">
      <resource path="update">
        <method name="POST">
          <request>
            <param href="#userParam"/>
            <param href="#passwordParam"/>
            <param href="#domainParam"/>
            <param href="#typeParam"/>
            <param href="#rdataParam"/>
            <param href="#ttlParam"/>
          </request>
        </method>
      </resource>
    </resource>
  </resources>
  <param name="user" type="xsd:string" style="query" required="true"
      id="userParam"/>
  <param name="password" type="xsd:string" style="query"
      required="true"
      id="passwordParam"/>
  <param name="domain" type="xsd:string" style="query" required="true"
      id="domainParam"/>
  <param name="type" type="xsd:string" style="query" id="typeParam"/>
  <param name="rdata" type="xsd:string" style="query" id="rdataParam"/>
  <param name="ttl" type="xsd:integer" style="query" id="ttlParam"/>
</application>
6. IANA Considerations

This document makes no requests of IANA.

7. Security Considerations

The request includes a clear text password and MUST be done over HTTPS or the password may be seen by an attacker and used to hijack the services.

If a user publishes the IP of their notebook computer, PDA, or smart phone as the move, it is likely that the IP address can be correlated to locations. By looking at the location over time for a specific user, it may be further possible to correlate that to an actual person. These attacks and implications to privacy are discussed in [dns-track].

8. Open Issues

Using HTTP Digest vs URL parameters.

Way to set the resource record to the IP address that the server got the request from.

9. Why not use DNS Dynamic Update?

TODO - lots of work is needed here.

RFC 2136 and the security for this provided by 3007, and the later DNSSEC RFCs provide a robust system for updating DNS that supports static shared secrets and asymmetric keys. Security working with asymmetric keys not easy but doing with static keys is vulnerable to offline attacks. Hard to do from Java script. Questions, any issues punching through NATs that have DNS ALGs with this? Hard to integrate with fine web security systems like openid. Trivial to implement this most web environments.

Questions about deployment success. When were things defined, what is the market choosing? Does it work?

Is the problem that we just need a simple open source library that does Dynamic Update?
10. Acknowledgements

Thanks to Frank Ellermann, Peter Koch, Stephane Bortzmeyer, Mark Baker, Patrik Faltstrom, and Julian Reschke.

11. References

11.1. Normative References


11.2. Informative References


Authors’ Addresses

Cullen Jennings
Cisco Systems
170 West Tasman Drive
Mailstop SJC-21/2
San Jose, CA  95134
USA
Phone:  +1 408 902-3341
Email:  fluffy@cisco.com

Tom Daly
Dynamic Network Services, Inc.

Email:  tom@dyn-inc.com

Jeremy Hitchcock
Dynamic Network Services, Inc.

Email:  jeremy@dyndns.com