Domain Name System Operations
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

A. Gavrichenkov
Qrator Labs

March 05, 2018

Domain Name System Service Application Programming Interface
draft-gavrichenkov-dnsop-dnssapi-00

Abstract

Managed DNS services are widely used to maintain DNS zones. Virtually all of them have an API of some sort, in most cases an XML-RPC or JSON-RPC API, while most of them lack the support of zone transfers. The latter is unlikely to change any time soon due to the reasons outlined below. This document describes a protocol, a common denominator of existing API protocols, that both a service provider and its customer can use to exchange information about DNS zones and policies.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on September 6, 2018.

Table of Contents

1. Introduction

Today, managed DNS services are a common solution for setting up and maintaining a DNS infrastructure for an enterprise. Those services often offer convenient functionality out of the box, e.g. failover, granular load balancing or geotargeting, while being more resilient to distributed denial-of-service attacks than a simple in-house solution could be.

However, the main challenge with managed DNS services is managing them. In case there's an update in the DNS setup, an enterprise would want it to be propagated to the managed service as soon as possible. However, existing mechanisms like zone transfer [RFC5936] or dynamic updates [RFC2136] are rarely implemented by managed DNS service providers, leaving an enterprise with an uncomfortable choice of either using a Web interface to manually set up zones and policies, or using an API of that provider.

There are reasons why existing mechanisms fail to gain popularity among service providers and their customers. First, zone transfer doesn't support virtually any of the features a customer might want from a service provider, except for a trivial name resolution. For instance, it is impossible to propagate a geography-based policy towards a service provider using zone transfer itself, with accordance to standard; this can be achieved ad hoc if both a customer and a provider agree on a particular zone naming policy, however, as this is not supported by an Internet standard, it makes changing a service provider or adding a new one a touch challenge.

Second, an enterprise which is using a managed DNS service might not be operating its own primary DNS server at all, sticking with simple deployment database exports. An XML-RPC or JSON-RPC API fits that model rather well, as handlers for those are highly likely to be implemented by a personnel quite familiar with the concepts of XML and/or JSON-RPC. However, implementing a binary protocol might be viewed as another challenge.

Next, both zone transfers and dynamic updates go in one direction, while an enterprise generally might want a feedback, including, but not limited to, traffic statistics overview, average response time, query type statistics, and so on. This is, once again, usually incorporated in a managed DNS service API.

However, the main issue with the latter is that there's currently no Internet standard providing a guidance for the API design. As the result, each DNS provider implements and maintains its own API, with its own naming schemes and type layouts, once again making migration from one provider to another - or operating more than one provider simultaneously - a challenge for network and system operations departments.

This might be viewed by some as a sort of a vendor lock-in, however, this issue alone is highly unlikely to really help retaining a customer who is somehow dissatisfied with the service and is eager to change the provider. What is beyond doubt is that a customer will just be further disappointed after they will face all the projected issues while moving to another service.

This way, it might be useful to agree on a common API protocol, JSON-RPC-based, with a built-in support for all the features offered by managed DNS services today, and extensible in order to add more features in future. The purpose of this document is to provide a description of such a protocol. This protocol might then be viewed as a guidance for new DNS providers which are going to implement their API, or for existing providers refactoring their code.

2. Functionality supported by managed DNS service providers

Here is the list of features implemented by managed DNS service providers (MDNSSP) today.

2.1. Failover

Enterprises are often in a high demand for online business continuency, and as the result, they opt for some redundancy. E.g. if they operate a Web site, they often have more than one server, on

more than one IP address, serving the Web content. There are mainly two options to implement that redundancy:

- Those servers may be put in an anycast IP prefix, announced from different locations, so that if a location goes down, its traffic is then served by nearest network locations

- Those servers may operate simultaneously, on a round-robin basis, all being put in a DNS A record entry.

The issue with the latter approach is that one has to set up monitoring and keep-alive checks of some sort to take a failing server out of round-robin as soon as possible. MDNSSP often offer convenient built-in features to do that.

2.2. Location-based DNS routing

Geography-based DNS routing, known also as geo-balancing, is a widely used method to reduce the latency between network clients and services by looking at the IP source of a DNS query and returning an answer with an IP address which is as close to a client as possible in terms of geolocation. The distance between a client and each in the set of servers may be measured in different ways, including looking at the country a source IP address belongs to, a region or city, or even comparing latitude and longitude.

However, due to routing policies of network operators and also due to the reported inaccuracy of regional internet registries' databases (which are the only officially recognized source of the mapping between IP addresses and countries and geographic regions), there might be latency issues now with geography-based DNS routing. Some MDNSSP handle that by allowing more specific policies to be set up, e.g. ASN-based or prefix-based policies.

2.3. Firewalling

Firewall access rules might be viewed as a subset of location-based policies, except for a simpler policy of just dropping the traffic instead of processing it. However, sometimes further requirements may take place, e.g. forcing a challenge towards a source IP address, and so on. Those features are a subject of a different extension than location-based routing, being applied before it.

2.4. Load balancing

There are a lot of things a DNS server can do to balance traffic towards a set of servers. The simplest example would be shuffling answers based on their weight.

2.5. Rate limiting

An MDNSSP may limit the amount of requests coming towards a single server by returning intentionally wrong reponse to an A query, e.g. NXDOMAIN. This might help to keep a server running in case of a sudden traffic spike.

The exact amount of queries triggering that condition must be specified as an argument during the setup.

2.6. Statistics

Generally, an MDNSSP offers metrics regarding the overall inbound and outbound network traffic, query count, average and/or median response time, and all or some of this data for different query names, types, response codes and so on.

3. General policy on additional extensions

The API is designed to be extensible. An MDNSSP SHOULD implement functionality in a way specified by this document in case this functionality can be handled by methods described in this specification. However, an MDNSSP MAY implement its own private extension if the standard functionality doesn't fit their needs.

An extension for the DNSSAPI protocol must either follow the naming structure for the private extensions' domain or use an IANA-allocated extension name.

4. DNSSAPI protocol specification

...

5. Normative References

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <https://www.rfc-editor.org/info/rfc2136>.

[RFC5936] Lewis, E. and A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, DOI 10.17487/RFC5936, June 2010, <https://www.rfc-editor.org/info/rfc5936>.

Author's Address

Artyom Gavrichenkov
Qrator Labs

Email: ag@qrator.net