Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 15, 2019

S. Cheshire
Apple Inc.
T. Lemon
Nibbhaya Consulting

July 14, 2018

Service Registration Protocol for DNS-Based Service Discovery
draft-sctl-service-registration-02

Abstract

The DNS-SD Service Registration Protocol uses the standard DNS Update mechanism to enable DNS-Based Service Discovery using only unicast packets. This eliminates the dependency on Multicast DNS as the foundation layer, which greatly improves scalability and improves performance on networks where multicast service is not an optimal choice, particularly 802.11 (Wi-Fi) and 802.15.4 (IoT) networks. DNS-SD Service registration uses public keys and SIG(0) to allow services to defend their registrations against attack.

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 January 15, 2019.

Copyright Notice

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

1. Introduction

DNS-Based Service Discovery [RFC6763] is a component of Zero Configuration Networking [RFC6760] [ZC] [I-D.cheshire-dnssd-roadmap].

This document describes an enhancement to DNS-Based Service Discovery [RFC6763] that allows services to automatically register their services using the DNS protocol rather than using mDNS. There is already a large installed base of DNS-SD clients that can do service discovery using the DNS protocol. This extension makes it much easier to take advantage of this existing functionality.

This document is intended for three audiences: implementors of software that provides services that should be advertised using DNS-SD, implementors of DNS servers that will be used in contexts where DNS-SD registration is needed, and administrators of networks where DNS-SD service is required. The document is intended to provide sufficient information to allow interoperable implementation of the registration protocol.

DNS-Based Service Discovery (DNS-SD) allows services to advertise the fact that they provide service, and to provide the information required to access that service. Clients can then discover the set of services of a particular type that are available. They can then select a service from among those that are available and obtain the information required to use it.

The DNS-SD Service Registration protocol, described in this document, provides a reasonably secure mechanism for publishing this information. Once published, these services can be readily discovered by clients using standard DNS lookups.

In the DNS-Based Service Discovery specification [RFC6763] Section 10 "Populating the DNS with Information" briefly discusses ways that services can publish their information in the DNS namespace. In the case of Multicast DNS [RFC6762], it allows services to publish their information on the local link, using names in the ".local" namespace, which makes their services directly discoverable by peers attached to that same local link.

RFC6763 also allows clients to discover services using the DNS protocol [RFC1035]. This can be done by having a system administrator manually configure service information in the DNS, but

manually populating DNS authoritative server databases is costly and potentially error-prone, and requires a knowledgable network administrator. Consequently, although all DNS-SD client implementations of which we are aware support DNS-SD using DNS queries, in practice it is used much less frequently than mDNS. The Discovery Proxy [I-D.ietf-dnssd-hybrid] provides one way to automatically populate the DNS namespace, but is only appropriate on networks where services are already advertised using mDNS. This document describes a solution more suitable for networks where multicast is inefficient, or undesirable for other reasons, by supporting both offering of services, and discovery of services, using unicast.

2. Service Registration Protocol

Services that implement the DNS-SD Service Registration Protocol use DNS Update [RFC2136] [RFC3007] to publish service information in the DNS. Two variants exist, one for full-featured devices, and one for devices designed for "Constrained-Node Networks" [RFC7228].

Full-featured devices are either configured manually, or use the "dr._dns-sd._udp" query [RFC6763] to learn the default registration domain from the network. Using the chosen service registration domain, full-featured devices construct the names of the SRV, TXT, and PTR records describing their service(s). For these names they then discover the zone apex of the closest enclosing DNS zone using SOA queries [I-D.ietf-dnssd-push]. Having discovered the enclosing DNS zone, they query for the "_dns-update._udp<zone>" SRV record to discover the server to which they should send DNS updates.

For devices designed for "Constrained-Node Networks" [RFC7228] some simplifications are used. Instead of being configured with (or discovering) the service registration domain, the (proposed) special use domain name [RFC6761] "services.arpa" is used. Instead of learning the server to which they should send DNS updates, a fixed IPv6 anycast address is used (value TBD). It is the responsibility of a "Constrained-Node Network" supporting DNS-SD Service Registration Protocol to provide appropriate anycast routing to deliver the DNS updates to the appropriate server. It is the responsibility of the DNS-SD Service Registration server on a "Constrained-Node Network" to handle the updates appropriately. In some network environments, updates may be accepted directly into a local "services.arpa" zone, which has only local visibility. In other network environments, updates for names ending in "services.arpa" may be rewritten internally to names with broader visibility.

The reason for these different assumptions is that "Constrained-Node Networks" generally require special egress support, and Anycast packets captured at the "Constrained-Node Network" egress can be assumed to have originated locally. Low-power devices that typically use "Constrained-Node Networks" may have very limited battery power. The additional DNS lookups required to discover a registration server and then communicate with it will increase the power required to advertise a service; for low-power devices, the additional flexibility this provides does not justify the additional use of power.

General networks have the potential to have more complicated topologies at the Internet layer, which makes anycast routing more difficult. Such networks may or may not have the infrastructure required to route anycast to a server that can process it. However, they can be assumed to be able to provide registration domain discovery and routing. By requiring the use of TCP, the possibility of off-network spoofing is eliminated.

We will discuss several parts to this process: how to know what to publish, how to know where to publish it (under what name), how to publish it, how to secure its publication, and how to maintain the information once published.

2.1. What to publish

We refer to the message that services using the DNSSD Registration Protocol send as a Registration. Three types of updates appear in a Registration: Service Discovery records, Service Description records, and Host Description records.

- Service Discovery records are one or more PTR RRs, mapping from the generic service type (or subtype) to the specific Service Instance Name.

- Service Description records are exactly one SRV RR, and one or more TXT RRs, both with the same name, the Service Instance Name ([RFC6763] section 4.1). In principle Service Description records can include other record types, with the same Service Instance Name, though in practice they rarely do. The Service Instance Name MUST be referenced by one or more Service Discovery PTR records, unless it is a placeholder service registration for an intentionally non-discoverable service name.

- The Host Description records for a service are a KEY RR, used to claim exclusive ownership of the service registration, and one or more RRs of type A or AAAA, giving the IPv4 or IPv6 address(es) of the host where the service resides.

RFC 6763 describes the details of what each of these types of updates contains and is the definitive source for information about what to publish; the reason for mentioning it here is to provide the reader with enough information about what will be published that the service registration process can be understood at a high level without first learning the full details of DNS-SD. Also, the "Service Instance Name" is an important aspect of first-come, first-serve naming, which we describe later on in this document.

2.2. Where to publish it

Multicast DNS uses a single namespace, ".local", which is valid on the local link. This convenience is not available for DNS-SD using the DNS protocol: services must exist in some specific unicast namespace.

As described above, full-featured devices are responsible for knowing in what domain they should register their services. Devices made for "Constrained-Node Networks" register in the (proposed) special use domain name [RFC6761] "services.arpa", and let the DNS-SD Service Registration server handle rewriting that to a different domain if necessary.

2.3. How to publish it

It is possible to issue a DNS Update that does several things at once; this means that it’s possible to do all the work of adding a PTR resource record to the PTR RRset on the Service Name if it already exists, or creating one if it doesn’t, and creating or updating the Service Instance Name and Host Description in a single transaction.

A Registration is therefore implemented as a single DNS Update message that contains a service’s Service Discovery records, Service Description records, and Host Description records.

Updates done according to this specification are somewhat different than regular DNS Updates as defined in RFC2136. RFC2136 assumes that updating is a fairly heavyweight process, so you might first attempt to add a name if it doesn’t exist, and then in a second message update the name if it does exist but matches certain preconditions. Because the registration protocol uses a single transaction, some of this adaptability is lost.

In order to allow updates to happen in a single transaction, Registrations do not include update constraints. The constraints specified in Section 2.4.2 are implicit in the processing of

Registrations, and so there is no need for the service sending the Registration to put in any explicit constraints.

2.3.1. How DNS-SD Service Registration differs from standard RFC2136 DNS Update

DNS-SD Service Registration is based on standard RFC2136 DNS Update, with some differences:

- It implements first-come first-served name allocation, protected using SIG(0).
- It enforces policy about what updates are allowed.
- It optionally performs rewriting of "services.arpa" to some other domain.
- It optionally performs automatic population of the address-to-name reverse mapping domains.
- A DNS-SD Service Registration server is not required to implement general DNS Update prerequsite processing.
- Simplified clients are allowed to send updates to an anycast address, for names ending in "services.arpa"

2.3.2. Testing using standard RFC2136-compliant servers

It may be useful to set up a DNS server for testing that does not implement the Registration protocol. This can be done by configuring the server to listen on the anycast address, or advertising it in the _dns-update._udp SRV record. It must be configured to be authoritative for "services.arpa", and to accept updates from hosts on local networks for names under "services.arpa" without authentication.

A server configured in this way will be able to successfully accept and process Registrations from services that send Registrations. However, no constraints will be applied, and this means that the test server will accept internally inconsistent Registrations, and will not stop two Registrations, sent by different services, that claim the same name(s), from overwriting each other.

2.3.3. How to allow services to update standard RFC2136-compliant servers

Ordinarily Registrations will fail when sent to any non-Registration Protocol server because the zone being updated is "services.arpa", and no DNS server that is not a Registration Protocol server should normally be configured to be authoritative for "services.arpa". Therefore, a service that sends a Registration can tell that the receiving server does not support the Registration Protocol, but does support RFC2136, because the RCODE will either be NOTZONE, NOTAUTH or REFUSED, or because there is no response to the update request (when using the anycast address)

In this case a service MAY attempt to register itself using regular RFC2136 DNS updates. To do so, it must discover default registration zone and the DNS server designated to receive updates for that zone, as described earlier using the _dns-update._udp SRV record. It can then make the update using the port and host pointed to by the SRV record, and should use appropriate constraints to avoid overwriting competing records. Such updates are out of scope for the DNSSD Registration Protocol, and a service that implements the DNSSD Registration Protocol MUST first attempt to use the Registration Protocol to register itself, and should only attempt to use RFC2136 backwards compatibility if that fails.

2.4. How to secure it

Traditional DNS update is secured using the TSIG protocol, which uses a secret key shared between the client (which issues the update) and the server (which authenticates it). This model does not work for automatic service registration.

The goal of securing the DNS-SD Registration Protocol is to provide the best possible security given the constraint that service registration has to be automatic. It is possible to layer more operational security on top of what we describe here, but what we describe here improves upon the security of mDNS. The goal is not to provide the level of security of a network managed by a skilled operator.

2.4.1. First-Come First-Served Naming

First-Come First-Serve naming provides a limited degree of security: a service that registers its service using DNS-SD Registration protocol is given ownership of a name for an extended period of time based on the key used to authenticate the DNS Update. As long as the registration service remembers the Service Instance Name and the key

used to register that Service Instance Name, no other service can add or update the information associated with that Service Instance Name.

2.4.1.1. Service Behavior

The service generates a public/private key pair. This key pair MUST be stored in stable storage; if there is no writable stable storage on the client, the client MUST be pre-configured with a public/private key pair that can be used.

When sending DNS updates, the service includes a KEY record containing the public portion of the key in each Host Description update. The update is signed using SIG(0), using the private key that corresponds to the public key in the KEY record. The lifetimes of the records in the update is set using the EDNS(0) Update Lease option.

The lifetime of the DNS-SD PTR, SRV, A, AAAA and TXT records [RFC6763] is typically set to two hours. This means that if a device is disconnected from the network, it does not appear in the user interfaces of devices looking for services of that type for too long.

However, the lifetime of its KEY record should be set to a much longer time, typically 14 days. The result of this is that even though a device may be temporarily unplugged, disappearing from the network for a few days, it makes a claim on its name that lasts much longer.

This way, even if a device is unplugged from the network for a few days, and its services are not available for that time, no other rogue device can come along and immediately claim its name the moment it disappears from the network. In the event that a device is unplugged from the network and permanently discarded, then its name is eventually cleaned up and made available for re-use.

2.4.2. Registration Server Behavior

The Registration server checks each update in the Registration to see that it contains a Service Discovery update, a Service Description update, and a Host Description update.

An update is a Service Discovery update if it contains

- exactly one RRset update,
- which is for a PTR RR,
- which points to a Service Instance Name
- for which an update is present in the Registration.

An update is a Service Description update if, for the appropriate Service Instance Name, it contains

- exactly one "Delete all RRsets from a name" update,
- exactly one SRV RRset update,
- one or more TXT RRset updates,
- and the target of the SRV record update references a hostname for which there is a Host Description update in the Registration.

An update is a Host Description update if, for the appropriate hostname, it contains

- exactly one "Delete all RRsets from a name" update,
- A or AAAA RR update(s)
- a KEY RR update that adds a KEY RR that contains the public key corresponding to the private key that was used to sign the message,
- there is a Service Instance Name update in the Registration that updates an SRV RR so that it points to the hostname being updated by this update.

A Registration MUST include at least one Service Name update, at least one Service Description update, and exactly one Host Description update. An update message that does not is not a Registration. An update message that contains any other updates, or any update constraints, is not a Registration. Such messages should either be processed as regular RFC2136 updates, including access control checks and constraint checks, if supported, or else rejected with RCODE=REFUSED.

Note that if the definitions of each of these update types are followed carefully, this means that many things that look very much like Registrations nevertheless are not. For example, a Registration that contains an update to a Service Name and an update to a Service Instance Name, where the Service Name does not reference the Service Instance Name, is not a valid Registration message, but may be a valid RFC2136 update.

Assuming that an update message has been validated with these conditions and is a valid Registration, the server checks that the name in the Host Description update exists. If so, then the server checks to see if the KEY record on the name is the same as the KEY record in the update. If it is not, then the server MUST reject the Registration with the YXDOMAIN RCODE.

Otherwise, the server validates the update using SIG(0) on the public key in the KEY record of the Host Description update. If the validation fails, the server MUST reject the rejectration rejected

with the REFUSED RCODE. Otherwise, the update is considered valid and authentic, and is processed according to the method described in RFC2136. The status that is returned depends on the result of processing the update.

The server MAY add a Reverse Mapping that corresponds to the Host Description. This is not required because the Reverse Mapping serves no protocol function, but it may be useful for debugging, e.g. in annotating network packet traces or logs.

The server MAY apply additional criteria when accepting updates. In some networks, it may be possible to do out-of-band registration of keys, and only accept updates from pre-registered keys. In this case, an update for a key that has not been registered should be rejected with the REFUSED RCODE.

There are at least two benefits to doing this rather than simply using normal SIG(0) DNS updates. First, the same registration protocol can be used in both cases, so both use cases can be addressed by the same service implementation. Second, the registration protocol includes maintenance functionality not present with normal DNS updates.

Note that the semantics of using the Registration Protocol in this way are different than for typical RFC2136 implementations: the KEY used to sign the update in the Registration Protocol only allows the client to update records that refer to its Host Description. RFC2136 implementations do not normally provide a way to enforce a constraint of this type.

The server may also have a dictionary of names or name patterns that are not permitted. If such a list is used, updates for Service Instance Names that match entries in the dictionary are rejected with YXDOMAIN.

2.5. TTL Consistency

All RRs within an RRset are required to have the same TTL (Clarifications to the DNS Specification [RFC2181], Section 5.2). In order to avoid inconsistencies, the Registration Protocol places restrictions on TTLs sent by services and requires that Registration Protocol Servers enforce consistency.

Services sending Registrations MUST use consistent TTLs in all RRs within the Registration.

Registration Protocol servers MUST check that the TTLs for all RRs within the Registration are the same. If they are not, the Registration MUST be rejected with a REFUSED RCODE.

Additionally, when adding RRs to an RRset, for example when processing Service Discovery records, the server MUST use the same TTL on all RRs in the RRset. How this consistency is enforced is up to the implementation.

2.6. Maintenance

2.6.1. Cleaning up stale data

Because the DNS-SD registration protocol is automatic, and not managed by humans, some additional bookkeeping is required. When an update is constructed by the client, it MUST include include an EDNS(0) Update Lease Option [I-D.sekar-dns-ul]. The Update Lease Option contains two lease times: the Update Lease Time and the Instance Lease Time.

These leases are promises, similar to DHCP leases [RFC2131], from the client that it will send a new update for the service registration before the lease time expires. The Update Lease time is chosen to represent the time after the update during which the registered records other than the KEY record should be assumed to be valid. The Instance Lease time represents the time after the update during which the KEY record should be assumed to be valid.

The reasoning behind the different lease times is discussed in the section on first-come, first-served naming Section 2.4.1. DNS-SD Registration Protocol servers may be configured with limits for these values. A default limit of two hours for the Update Lease and 14 days for the SIG(0) KEY are currently thought to be good choices. Clients that are going to continue to use names on which they hold leases should update well before the lease ends, in case the registration service is unavailable or under heavy load.

The Registration Protocol server MUST include an EDNS(0) Update Lease option in the response if the lease time proposed by the service has been shortened. The service MUST check for the EDNS(0) Update Lease option in the response and MUST use the lease times from that option in place of the options that it sent to the server when deciding when to update its registration.

Clients should assume that each lease ends N seconds after the update was first transmitted, where N is the lease duration. Servers should assume that each lease ends N seconds after the update that was successfully processed was received. Because the server will always

receive the update after the client sent it, this avoids the possibility of misunderstandings.

DNS-SD Registration Protocol servers MUST reject updates that do not include an EDNS(0) Update Lease option. Dual-use servers MAY accept updates that don't include leases, but SHOULD differentiate between DNS-SD registration protocol updates and other updates, and MUST reject updates that are known to be DNS-SD Registration Protocol updates if they do not include leases.

2.6.2. Sleep Proxy

Another use of Service Registration Protocol is for devices that sleep to reduce power consumption.

In this case, in addition to the DNS Update Lease option [I-D.sekar-dns-ul] described above, the device includes an EDNS(0) OWNER Option [I-D.cheshire-edns0-owner-option].

The EDNS(0) Update Lease option constitutes a promise by the device that it will wake up before this time elapses, to renew its registration and thereby demonstrate that it is still attached to the network. If it fails to renew the registration by this time, that indicates that it is no longer attached to the network, and its registration (except for the KEY in the Host Description) should be deleted.

The EDNS(0) OWNER Option indicates that the device will be asleep, and will not be receptive to normal network traffic. When a DNS server receives a DNS Update with an EDNS(0) OWNER Option, that signifies that the Registration Protocol server should set up a proxy for any IPv4 or IPv6 address records in the DNS Update message. This proxy should send ARP or ND messages claiming ownership of the IPv4 and/or IPv6 addresses in the records in question. In addition, proxy should answer future ARP or ND requests for those IPv4 and/or IPv6 addresses, claiming ownership of them. When the DNS server receives a TCP SYN or UDP packet addressed to one of the IPv4 or IPv6 addresses for which it proxying, it should then wake up the sleeping device using the information in the EDNS(0) OWNER Option. At present version 0 of the OWNER Option specifies the "Wake-on-LAN Magic Packet" that needs to be sent; future versions could be extended to specify other wakeup mechanisms.

Note that although the authoritative DNS server that implements the DNSSD Service Registration Protocol function need not be on the same link as the sleeping host, the Sleep Proxy must be on the same link.

3. Security Considerations

DNS-SD Service Registration Protocol updates have no authorization semantics other than first-come, first-served. This means that if an attacker from outside of the administrative domain of the server knows the server's IP address, it can in principle send updates to the server that will be processed successfully. Servers should therefore be configured to reject updates from source addresses outside of the administrative domain of the server.

For Anycast updates, this validation must be enforced by every router that connects the CDN to the unconstrained portion of the network. For TCP updates, the initial SYN-SYN+ACK handshake prevents updates being forged from off-network. In order to ensure that this handshake happens, Service Discovery Protocol servers MUST NOT accept 0-RTT TCP payloads.

Note that these rules only apply to the validation of DNS-SD registration protocol updates. A server that accepts updates from DNS-SD registration protocol clients may also accept other DNS updates, and those DNS updates may be validated using different rules. However, in the case of a DNS service that accepts automatic updates, the intersection of the DNS-SD service registration update rules and whatever other update rules are present must be considered very carefully.

For example, a normal, authenticated RFC2136 update to any RR that was added using the Registration protocol, but that is authenticated using a different key, could be used to override a promise made by the registration protocol, by replacing all or part of the service registration information with information provided by a different client. An implementation that allows both kinds of updates should not allow updates to records added by Registrations using different authentication and authorization credentials.

4. Privacy Considerations

5. Acknowledgments

Thanks to Toke Hoeiland-Joergensen for a thorough technical review, to Tamara Kemper for doing a nice developmental edit, Tim Wattenberg for doing a service implementation at the Montreal Hackathon at IETF 102, and [...] more reviewers to come, hopefully.

6. References

6.1. Normative References

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/info/rfc6763>.

[I-D.sekar-dns-ul] Sekar, K., "Dynamic DNS Update Leases", draft-sekar-dns-ul-01 (work in progress), August 2006.

6.2. Informative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <https://www.rfc-editor.org/info/rfc2131>.

[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>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/info/rfc2181>.

[RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <https://www.rfc-editor.org/info/rfc2931>.

[RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <https://www.rfc-editor.org/info/rfc3007>.

[RFC3152] Bush, R., "Delegation of IP6.ARPA", BCP 49, RFC 3152, DOI 10.17487/RFC3152, August 2001, <https://www.rfc-editor.org/info/rfc3152>.

[RFC6760] Cheshire, S. and M. Krochmal, "Requirements for a Protocol to Replace the AppleTalk Name Binding Protocol (NBP)", RFC 6760, DOI 10.17487/RFC6760, February 2013, <https://www.rfc-editor.org/info/rfc6760>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <https://www.rfc-editor.org/info/rfc6761>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

[I-D.ietf-dnssd-hybrid] Cheshire, S., "Discovery Proxy for Multicast DNS-Based Service Discovery", draft-ietf-dnssd-hybrid-08 (work in progress), March 2018.

[I-D.ietf-dnssd-push] Pusateri, T. and S. Cheshire, "DNS Push Notifications", draft-ietf-dnssd-push-14 (work in progress), March 2018.

[I-D.cheshire-dnssd-roadmap] Cheshire, S., "Service Discovery Road Map", draft-cheshire-dnssd-roadmap-01 (work in progress), March 2018.

[I-D.cheshire-edns0-owner-option] Cheshire, S. and M. Krochmal, "EDNS0 OWNER Option", draft-cheshire-edns0-owner-option-01 (work in progress), July 2017.

[ZC] Cheshire, S. and D. Steinberg, "Zero Configuration Networking: The Definitive Guide", O'Reilly Media, Inc. , ISBN 0-596-10100-7, December 2005.

Authors' Addresses

Stuart Cheshire
Apple Inc.
One Apple Park Way
Cupertino, California 95014
USA

Phone: +1 408 974 3207
Email: cheshire@apple.com

Ted Lemon
Nibbhaya Consulting
P.O. Box 958
Brattleboro, Vermont 05302
United States of America

Email: mellon@fugue.com